ARMY RESEARCH LABORATORY

A Simple Data Logging System for Ballistic Applications

by Thomas Kottke

ARL-TR-3853 **July 2006**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-TR-3853 **July 2006**

A Simple Data Logging System for Ballistic Applications

Thomas Kottke
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
July 2006	Final	1 October 2005–31 March 2006

4. TITLE AND SUBTITLE
A Simple Data Logging System for Ballistic Applications

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Thomas Kottke

5d. PROJECT NUMBER
AH80

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRD-ARL-WM-TE
Aberdeen Proving Ground, MD 21005-5066

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-3853

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
An economical and robust data logger is presented that is well suited to ballistic environments. This ballistic data recorder acquires two channels of analog data over a 0- to 5-V range, with acquisition times as short as 2.3 µs and record lengths as large as 0.5 MB per channel. The microcontroller-based architecture allows many data acquisition parameters such as rate, mode, triggering method, and record length to be selected and varied by the user. Onboard batteries and charging circuitry further enhance the data logger's applicability and flexibility.

Complete details of the ballistic data logger's hardware and software are presented in this report. A description of the hardware begins with a broad overview of the ballistic data logger's capabilities and method of operation and increases in complexity to provide complete electronic schematics, fabrication methods, and component procurement information. A complete listing of data logger software is provided with extensive documentation. The ballistic data logger's performance is verified with an example of acquired data.

15. SUBJECT TERMS
data, logger, recorder, ballistic

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Thomas Kottke
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL	70	19b. TELEPHONE NUMBER (*Include area code*) 410-278-2557

Contents

List of Figures

List of Tables

Acknowledgments

The author would like to thank R. Brian Leavy of the Armor Mechanics Branch for providing seminal funding for this project; J. Wayne Gardiner, retired, and Robert M. Reinsel of the Materials Application Branch for assistance with waterjet fabrication technology; Keith A. Mahan of the Survivability Concepts Branch for machining assistance; Jeffrey L. Cameron of the Survivability Concepts Branch for electronic fabrication assistance; and Sarah W. Kennedy, Barbara E. Ringers, Peter T. Bartkowski, and James Tardif for reviewing and improving the final manuscript.

INTENTIONALLY LEFT BLANK.

1. Introduction

The Survivability Concepts Branch of the Terminal Effects Division of the Weapons and Materials Research Directorate, U.S. Army Research Laboratory, has developed an economical and robust data logger for ballistic applications. This apparatus was originally designed to record in-flight signals from infrared proximity sensors. However, the data logger's microcontroller-based architecture allows it to be reprogrammed to function in a wide variety of applications and is therefore useful to the ballistics community in general.

This report presents a complete description of both the hardware and software that were developed for the ballistic data logger. First, the capabilities of the data logger are presented along with an outline of the method of operation. The hardware is then reviewed in detail including electronic schematics, fabrication methods, and component procurement information. All the code that drives the data logger microcontroller is discussed and listed with embedded comments. Finally, an example is presented of data that have been collected by the ballistic data logger to verify its performance.

2. Ballistic Data Logger Embodiment

2.1 Overview

This section presents a broad overview of the ballistic data logger hardware. Specifically, the data logger's capabilities are enumerated to allow prospective users the opportunity to decide if this device can satisfy their requirements. Its method of operation is also presented at a level that allows the user to apply the data logger in an effective manner. Subsequent sections delve into the full-blown gory electronic details for readers who may need to modify or reproduce this hardware.

This ballistic data logger is a highly portable, microcontroller-based, 2 channel, 8-bit data acquisition system that can measure signals over a 0- to 5-V range yielding a measurement resolution of less than 20 mV. The data acquisition time for two-channel operation can be as short as 2.3 μs. This means that a voltage measurement can be recorded on both channels every 2.3 μs. Each channel has 0.5 MB of storage memory. Thus, for the 2.3-μs minimum data acquisition period a total recording time in excess of 1.2 s is available. Since the ballistic data logger's operation is controlled by an onboard microcontroller, the data acquisition time can be set by the user to any value greater than or equal to 2.3 μs. Table 1 presents a range of data acquisition times and the corresponding total recording times. This table highlights the data logger's ability to also serve as a long-term monitoring device for scenarios that require modest data acquisition rates.

Table 1. Ballistic data logger total recording times for a variety of data acquisition times.

Data Acquisition Time	Total Recording Time	Data Acquisition Time	Total Recording Time
2.3 µs	1.21 s	10 ms	87.4 min
10 µs	5.24 s	100 ms	14.6 hr
100 µs	52.4 s	1 s	6.07 day
1 ms	8.74 min	10 s	60.7 day

In addition to the data acquisition rate, the microcontroller-based architecture of the ballistic data logger allows many other data acquisition parameters to be selected and varied by the user. Similar to the operation of an oscilloscope, the data acquisition mode can be chosen to be single-sweep or retriggered. Data initiation can be triggered by detection of an input signal above a preset threshold level or by detection of a digital synchronization pulse. The length of the data record can be reduced for shorter events or expanded to fill 1 MB of memory for single-channel operation. Indeed, the ballistic data logger's flexibility is primarily limited by the ingenuity and expertise of the microcontroller's programmer.

Once data has been recorded, it is transferred from the ballistic data logger to a personal computer (PC) through a standard RS232 serial port connection. The baud rate and other details of this serial port connection are again chosen by the user and programmed into the onboard microcontroller.

The ballistic data logger is powered by onboard batteries to enhance its ease of use, flexibility, and robustness. As presented, these batteries provide for more than 2 hr of data recording operation and more than 1 day of data memory retention. All the required battery charging circuitry is contained in the data logger hardware. Therefore, a simple external power supply can be used to recharge the batteries or power the data logger for applications where the required run time exceeds the battery lifetime. The data logger can be remotely activated using an internal electronic switch.

A single 3.8-in-diameter printed circuit board (PCB) is used to fabricate the ballistic data logger. The total thickness of the completed assembly is about 3/4 in; the mass is 60 g. Four mounting holes are provided that are sized for 6–32 screws and four pass holes are available to allow wires to be threaded through the data logger. A picture of the ballistic data logger is presented in figure 1. The data logger's capabilities and characteristics are summarized in table 2.

2.2 Electronic Circuitry

A block diagram of the ballistic data logger's major components and the interconnections between them is displayed in figure 2. Figure 3 illustrates a more detailed schematic diagram of the data logger's circuitry. In the following discussion, references to details of this schematic diagram are italicized for the reader's convenience.

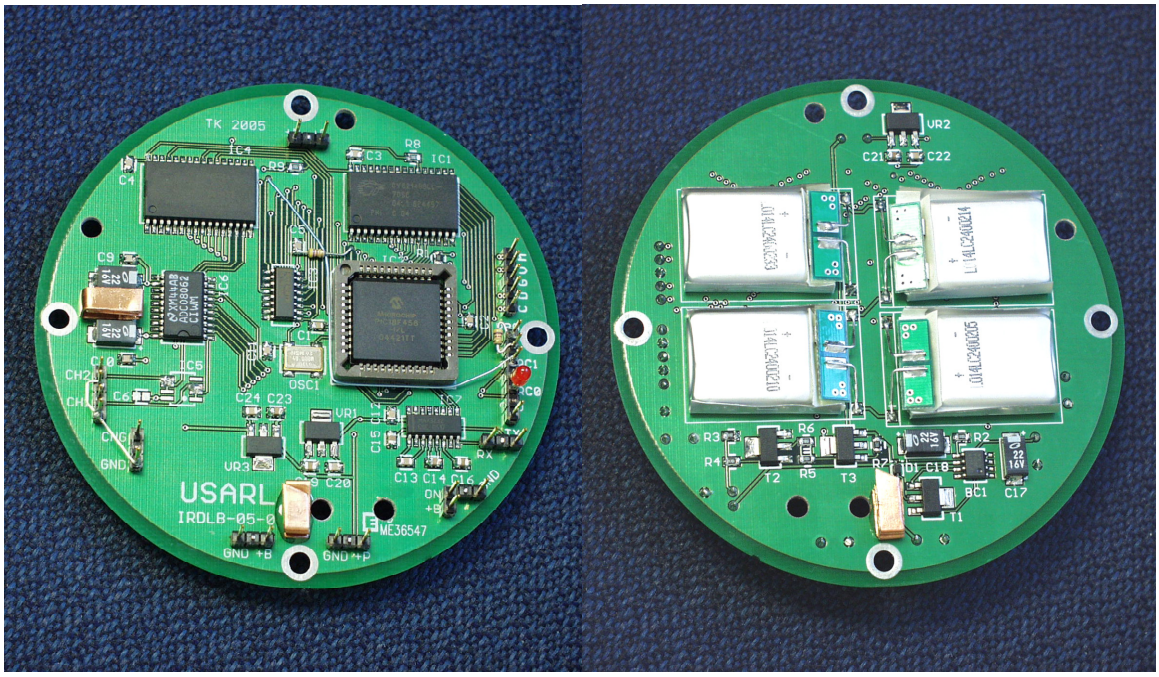

Figure 1. Photographs of top (left) and bottom (right) of ballistic data logger PCB.

Table 2. Ballistic data logger capabilities and characteristics.

Parameter	Value
Number of channels	2
Voltage measurement range	0 to 5 V
Number of data bits	8
Voltage measurement resolution	<20 mV
Memory per channel	0.5 MB
Acquisition mode	Single-sweep or retrigger
Triggering mode	Threshold or synch pulse
Data acquisition battery life	>2 hr
Memory retention battery life	>1 day
Activation control	Remote electronic switch
Diameter	3.8 in
Thickness	<3/4 in
Weight	60 g

The data logger is powered by onboard lithium polymer (LiPO) batteries. This family of rechargeable batteries provides high-power density, extended shelf life, and the potential for hundreds of recharging cycles with minimal degradation in cell capacity and voltage. However, these batteries must be recharged in a very specific manner in order to satisfy safety requirements (*1*). Specifically, the charging voltage must be limited to 4.2 V per battery cell in a series configuration and the charging current in amperes should be limited to the cell's capacity

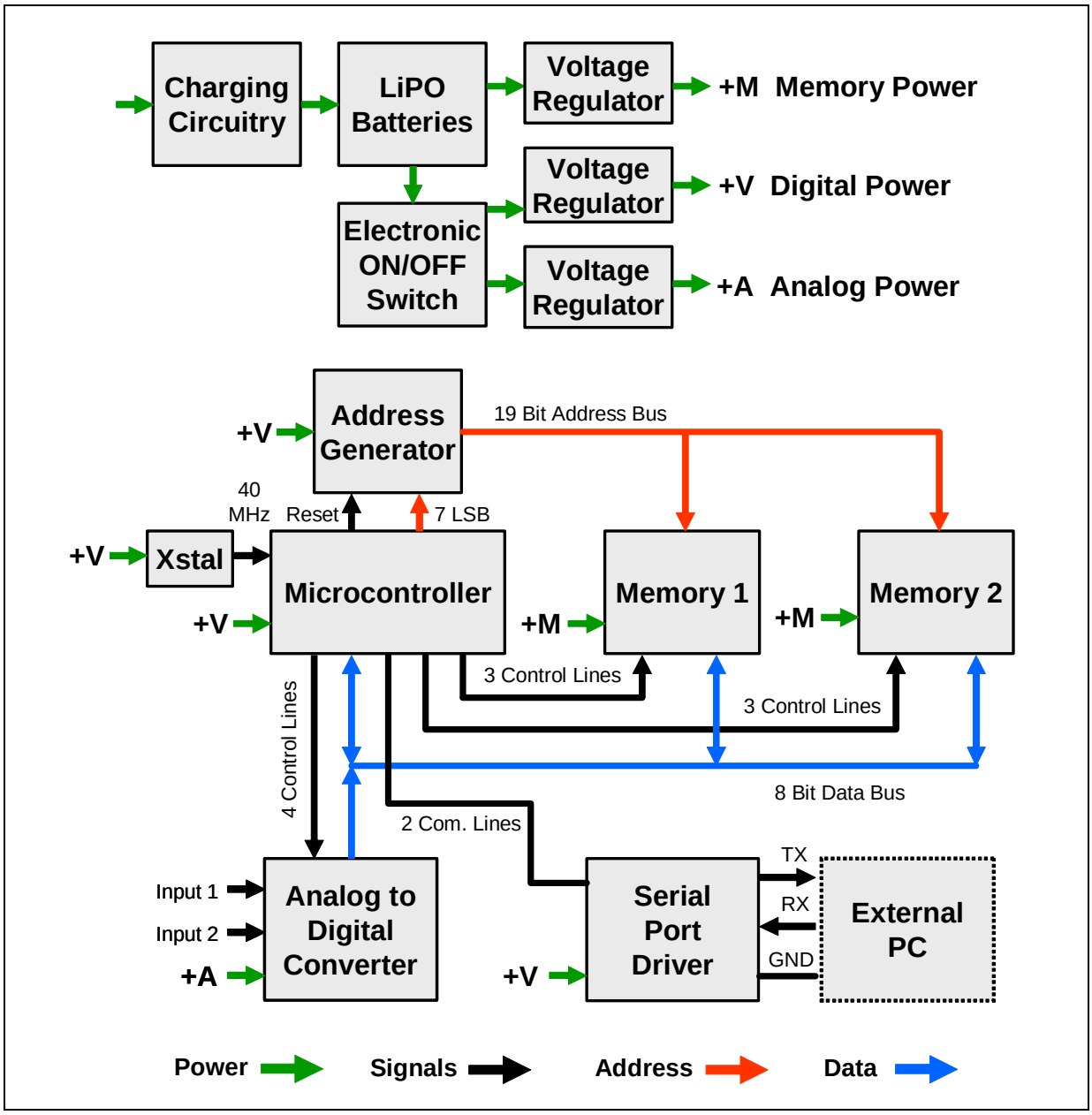

Figure 2. Block diagram of ballistic data logger hardware.

value in A·hr. Four Kokam KOK145T LiPO cells are used in each data logger.* These cells are configured as two parallel banks with each bank containing two cells in series. Therefore, the maximum charging voltage must be limited to 8.4 V. Each cell has a capacity value of 0.145 A·hr. When the two parallel battery banks are charged, only half the charging current will pass through each bank. Therefore, the total charging current should be limited to a maximum value of 0.290 A.

*FMA Direct, 5716A Industry Lane, Frederick, MD 21704, 800-343-2934, http://www.fmadirect.com.

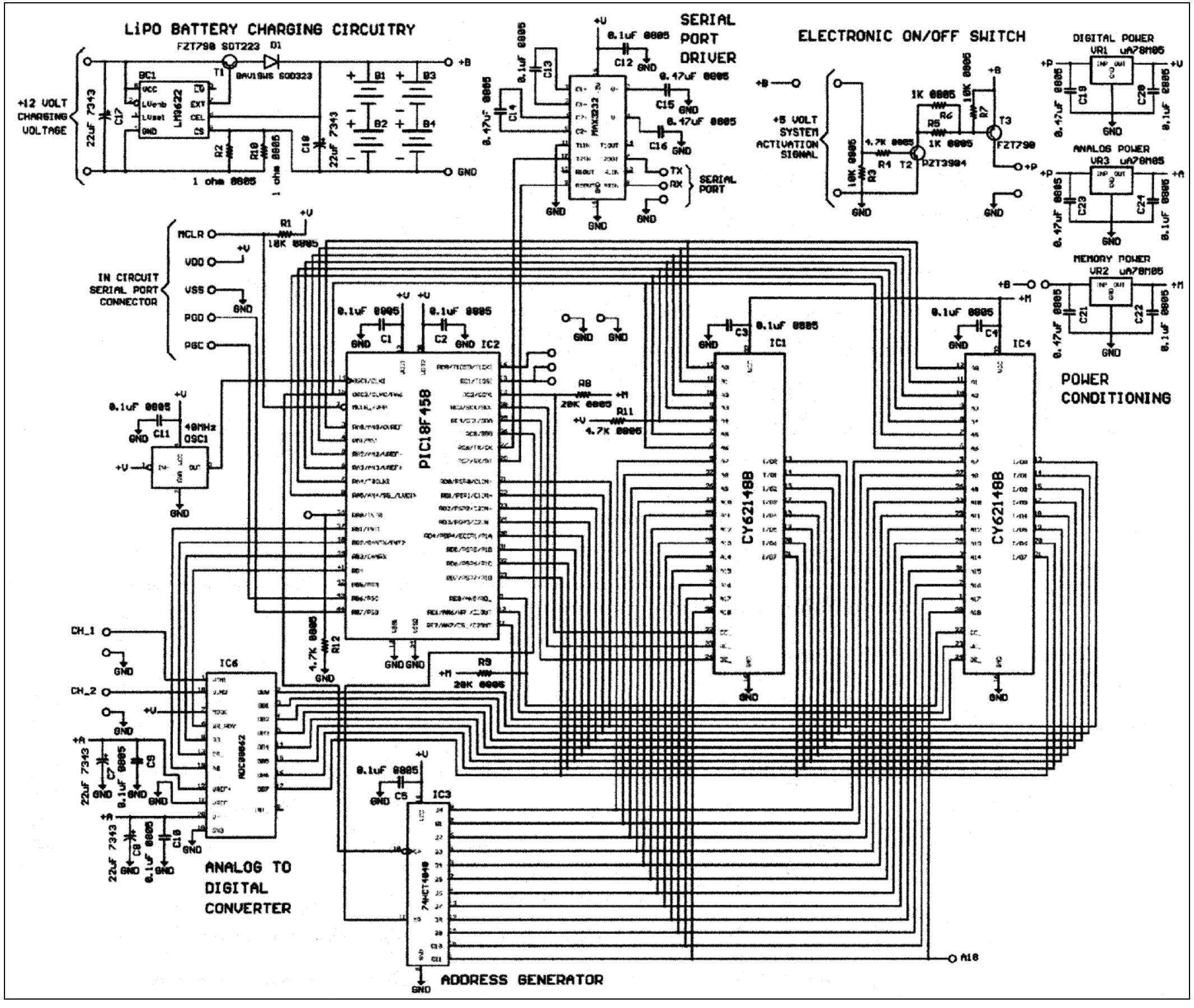

Figure 3. Schematic diagram of ballistic data logger hardware.

INTENTIONALLY LEFT BLANK.

Conveniently, the popularity of LiPO cells as a power source for cell phones has spawned the development of electronic components that are specifically designed to charge these cells in a safe and efficient manner. The ballistic data logger incorporates a National Semiconductor *LM3622M-8.4* lithium-ion battery charger controller to perform this function (*2*). This integrated circuit and the associated components that make up the charging circuitry are displayed in the top left section of figure 3. With this charging circuitry any 12- to 24-V, 0.5-A power supply can be used to charge the onboard LiPO battery pack to a maximum voltage of 8.4 V at a conservative current of 0.2 A in less than 2 hr. If this charging circuitry should fail, the LiPO battery pack can be charged manually using a voltage- and current-regulated power supply through the provided battery access terminals.

The energy from the LiPO battery pack is conditioned by three separate *uA78M05* positive-voltage regulators (*3*) to power the memory, digital, and analog portions of the ballistic data logger circuitry. Volatile memory is utilized in this data logger. This means the memory components must remain powered in order for stored data to be retained. Thus, the output from the LiPO battery pack, *+B*, is hardwired to the input of memory voltage regulator *VR2* when the data logger is operational. The regulated +5 V potential that *VR2* generates powers the memory components and is designated *+M*. For the digital and analog portions of the circuitry an electronic switch is provided that can be used to remotely turn these portions of the data logger on and off. As illustrated in the upper-right portion of figure 3, a 5-V control signal can turn on this electronic switch to pass the *+B* battery voltage through to the inputs of voltage regulators *VR1* and *VR3* as *+P*. If this electronic switch becomes nonfunctional, then these voltage regulators can be powered externally through the *+P* and *GND* access terminals. The output from voltage regulator *VR1* that powers the digital circuitry is designated *+V* and the output from voltage regulator *VR3* that powers the analog circuitry is designated *+A*. The analog and digital portions of the circuitry are provided with their own voltage regulators to isolate the noise sensitive analog components from the undesirable voltage spikes that are common on digital power lines. All the digital components include 0.1-µF bypass capacitors on their power lines to further mitigate these power spikes.

A *PIC18F458* high-performance enhanced FLASH microcontroller (*4*) is used to direct and coordinate the actions of the various components that make up the ballistic data logger. These actions include the initiation of the data acquisition process, the analog to digital conversion process, the storage of converted data, and the eventual transfer of acquired data. Each of these processes will now be considered in detail highlighting the components that are involved and how they are coordinated by the microcontroller.

Two methods are considered for initiating the data acquisition process. The first method involves the application of an external synchronization pulse. In this scheme the data acquisition code in the microcontroller is included in a high-priority interrupt routine. When triggered by the application of a suitable external trigger signal to the *RB0/INT0* line of the microcontroller, the interrupt data acquisition routine is run to completion before control is returned to the code

that was being executed prior to the interrupt. A second method for initiating the data acquisition process is to trigger when a particular input voltage exceeds some user-defined threshold value. In this scheme the data acquisition code begins with a free-running loop that repeatedly reads an applied analog voltage value and compares it to the specified threshold level. This free-running loop is repeated as long as the input value does not exceed the threshold value. Once the threshold value is exceeded the code branches out of the loop and the remainder of the data acquisition code is executed.

Analog input signals are digitized for subsequent storage by an *ADC08062* two-channel analog-to-digital converter (ADC) with internal sample and hold (*5*). This multiplexed ADC passes signals from either of the two analog inputs to a common digitizing unit that assures uniform digitization for both input channels. The *PIC18F458* microcontroller controls the operation of the *ADC08062* by manipulating four of its control lines. The *CS bar* chip select line is pulled low to activate the ADC for operation. A low signal to line *A0* selects analog input 1 for digitization while a high signal to *A0* selects analog input 2. Digital conversions are initiated by a falling signal on line *WR bar* and the digitized data is output from the ADC *DB#* data lines to the data bus by a low signal on *RD bar*.

The digitized data is stored in two *CY62148B* 512K word by 8-bit static random access memory (RAM) integrated circuits (*6*). Three control lines are used by the microcontroller to regulate the operation of each memory chip. A *CE bar* chip enable line is pulled low to select a memory chip for operation. Data is written to the memory chip by pulling the *CE bar* and the *WE bar* write enable lines low. Conversely, data is read from the memory chip by pulling the *CE bar* and the *OE bar* output enable lines low while forcing *WE bar* high. Read and write operations act on the memory location specified by the 19 address lines *A0–A18*. The seven lowest significance address bits, *A0–A6*, are generated by the microcontroller's port A digital output lines, *RA0–RA6*. A *CD74HCT4040* high-speed 12-stage binary counter (*7*) is used to generate the remaining address bits. The microcontroller resets this binary counter by momentarily pulling its *MR* master reset line high and the microcontroller clocks the counter's *CP* clock pulse line with the signal from address output line *RA6*. In essence, the binary counter's output lines are cascaded to the output from the microcontroller's port A lines to generate the complete memory chip address.

At this point, a couple of features about the microcontroller and the memory chips need to be highlighted. All of the microcontroller's port A lines, which are used to generate the seven lowest significance address bits, are transistor-transistor-logic (TTL) outputs, except for line *RA4*, which is configured as an open drain output. Therefore, in order to function as a TTL type output a 4.7-kΩ pull-up resistor, *R11*, is added to line *RA4*. The memory chips have an automatic power-down feature that can reduce power consumption by more than 99%. This feature is activated by deselecting the memory chip with a high level to the *CE bar* line. Pull up resistors *R8* and *R9* are attached to the *CE bar* lines of the memory chips to insure that this power saving feature is activated when the microcontroller is powered down.

After the data acquisition process is completed, data are continuously output through an onboard serial port that can be connected to an external PC for long-term data storage and analysis. Conveniently, the *PIC18F458* microcontroller includes a universal synchronous/asynchronous receiver/transmitter (USART) that provides the foundation for the serial port capability. However, this onboard USART transmits and receives TTL level signals that are not directly compatible with the RS232 serial port protocol (*8*). The proper RS232 signal levels are obtained by passing the microcontroller's USART signals through a *MAX3232* transceiver driver (*9*). This serial port driver and the associated circuitry are illustrated in the top middle portion of figure 3.

2.3 Electronics Fabrication

The ballistic data logger is assembled on a single four-layer PCB. Figures 4–7 individually display the design of the PCB's four layers. Figure 8 shows a combination of all four PCB layers with the ground planes and screen printing omitted to highlight the interconnections between layers. Surface mount devices (SMDs) are utilized in this device. These components are available from Digi-Key Corporation* and are listed in table 3 along with procurement information. The placement of the various SMD components on the data logger PCB is illustrated in figures 9 and 10.

Extra protection can be added to the ballistic data logger for applications in high-shock environments. An example of supplementary protection is presented in figure 11. Metal plates are added to the top and bottom surfaces of the data logger and the entire assembly is potted in high-thermal conductivity epoxy to yield a robust, monolithic structure. With this additional protection the data logger weighs 730 g and is 1 in thick.

*Digi-Key Corporation, 701 Brooks Ave. S., Thief River Falls, MN 56701-0677, 800-344-4539, http://www.digikey.com.

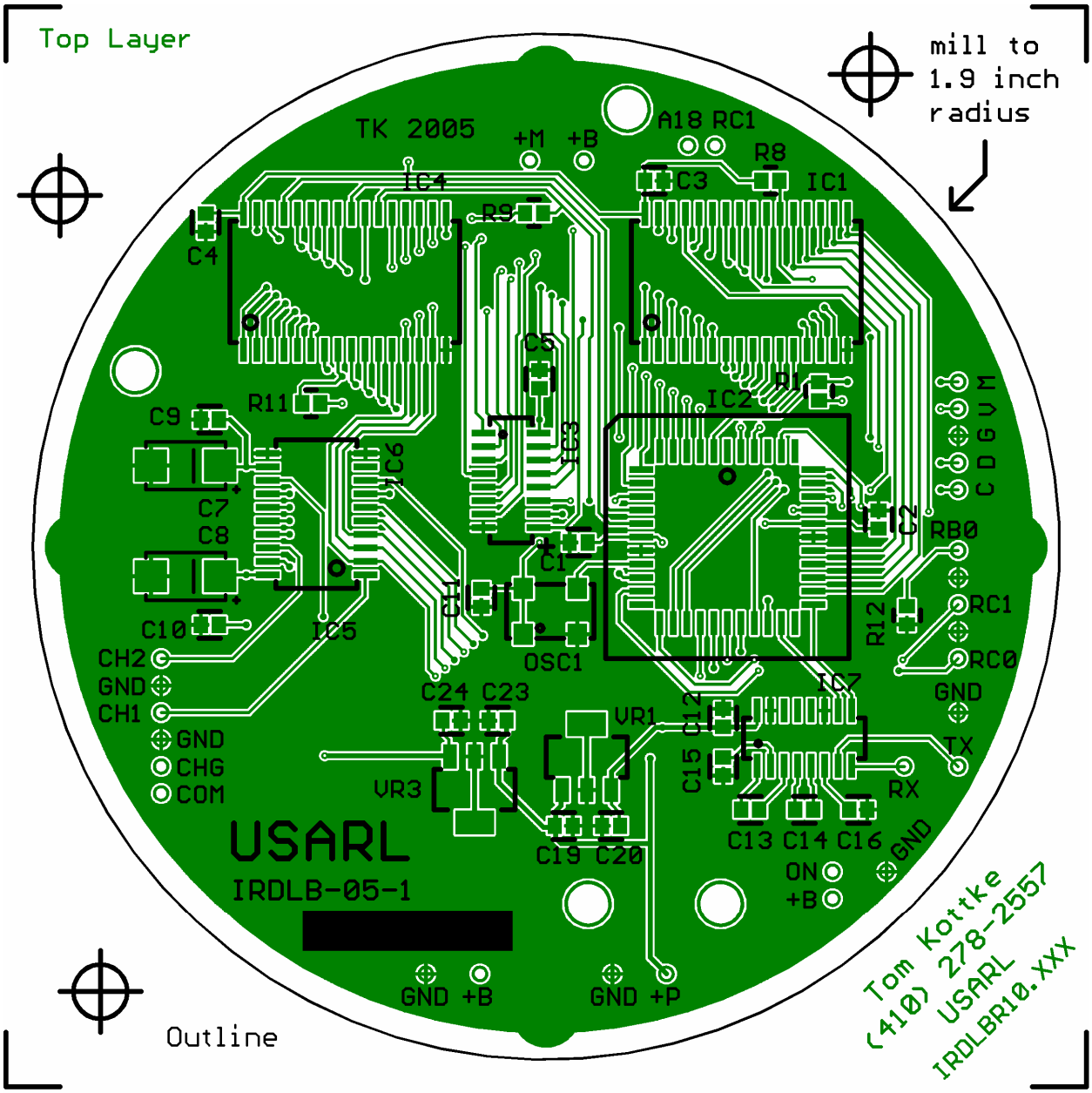

Figure 4. Design of the top or first layer of the ballistic data logger printed circuit board.

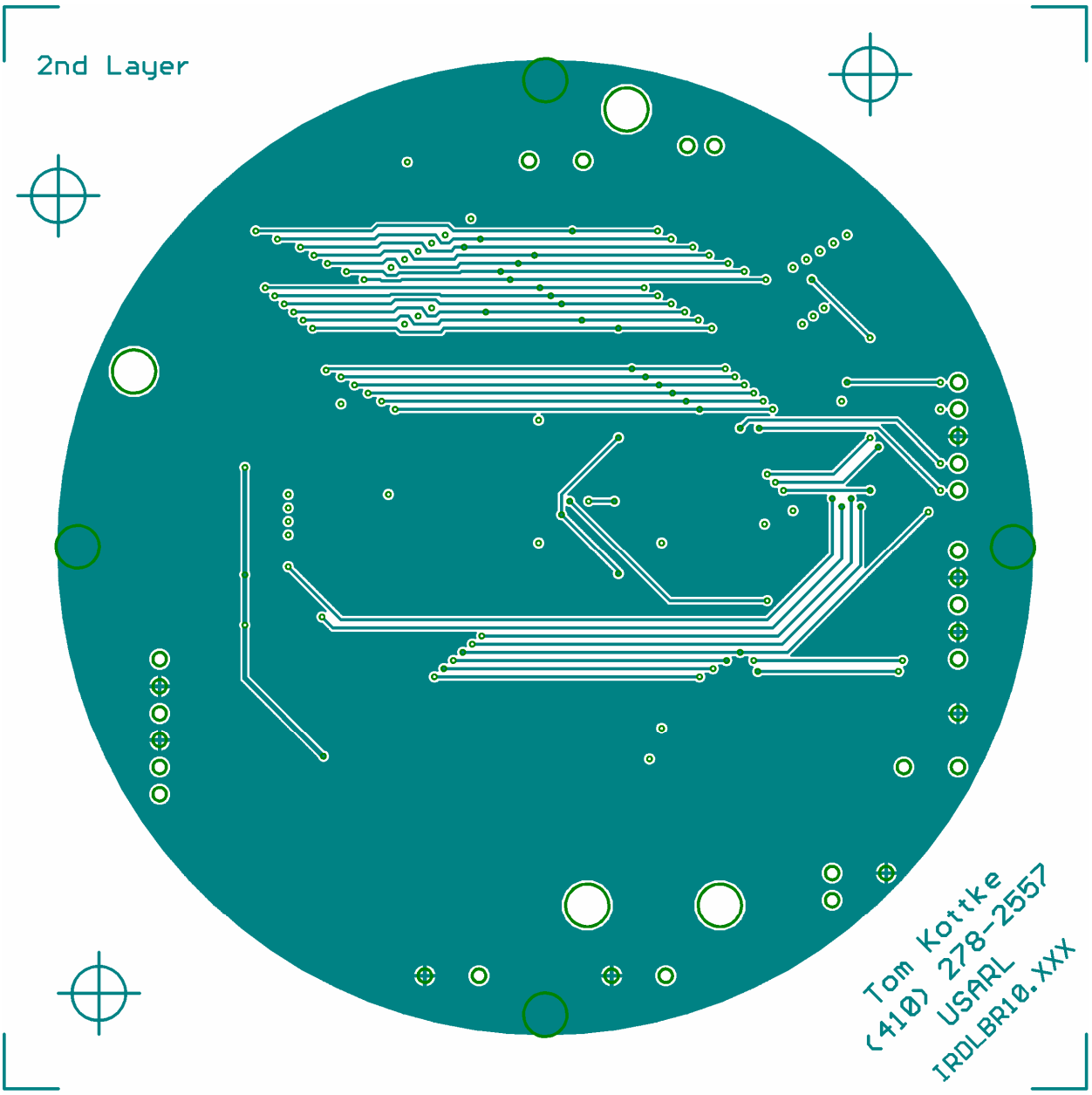

Figure 5. Design of the second layer of the ballistic data logger printed circuit board.

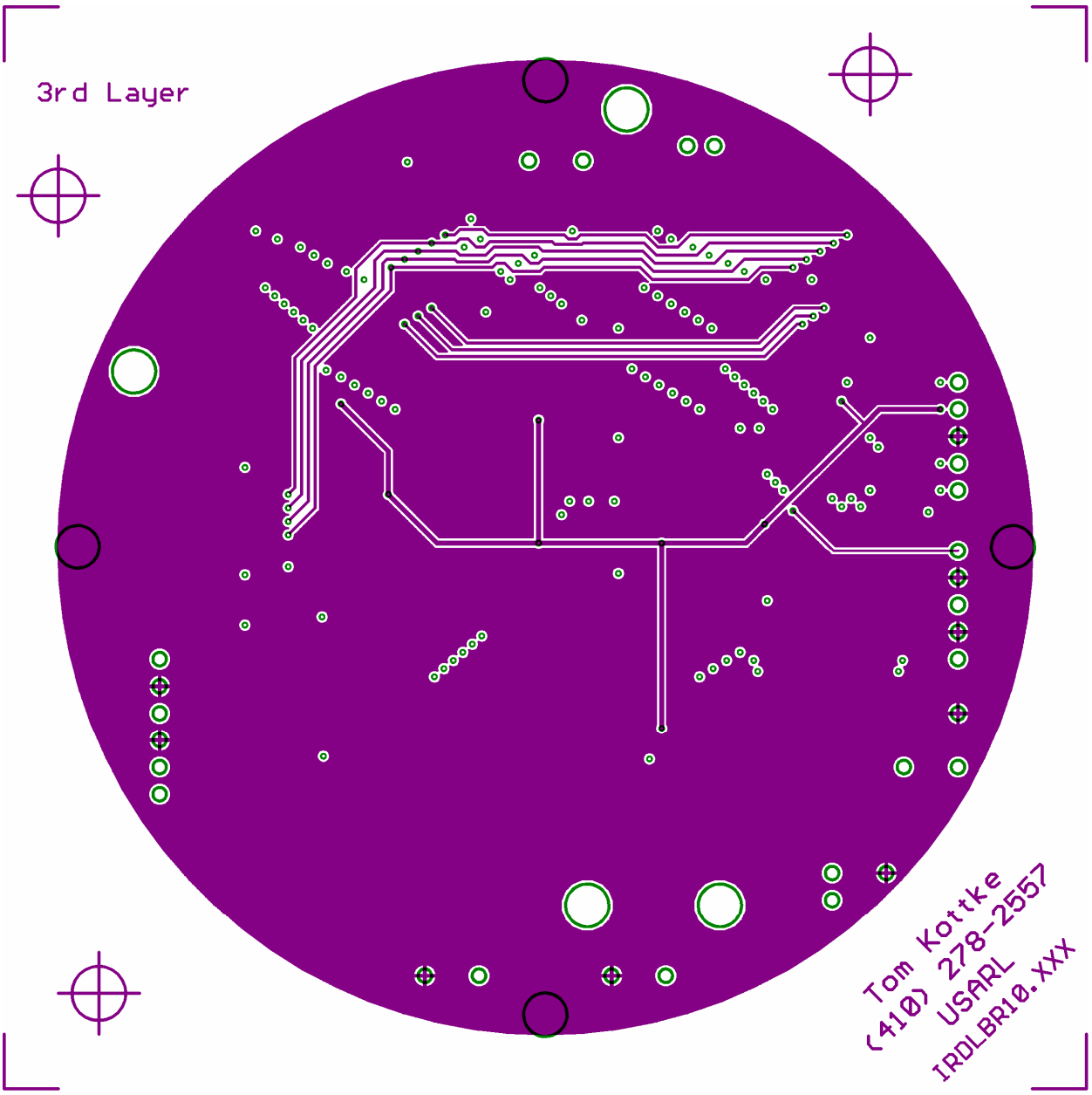

Figure 6. Design of the third layer of the ballistic data logger printed circuit board.

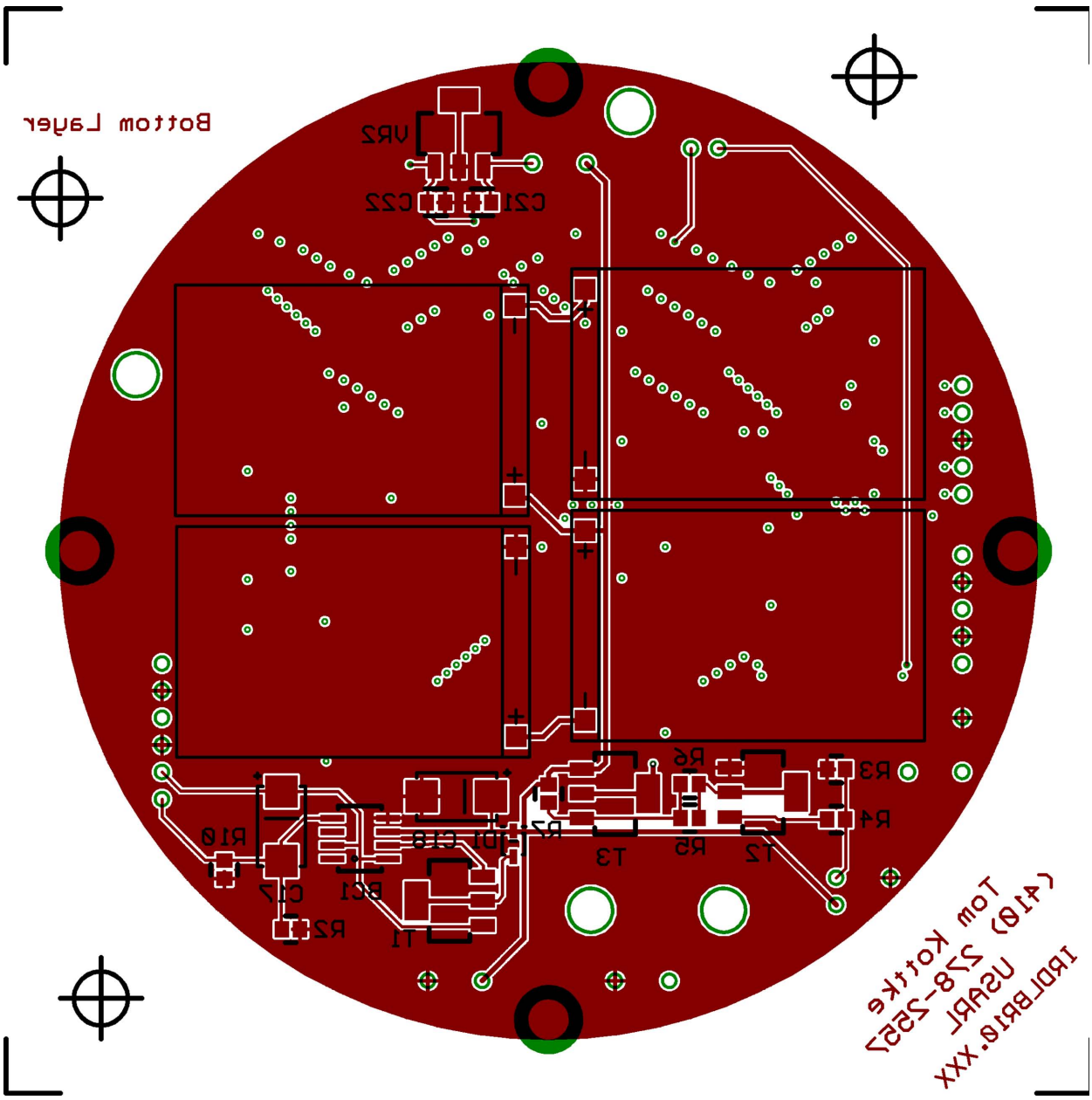

Figure 7. Design of the bottom or fourth layer of the ballistic data logger printed circuit board.

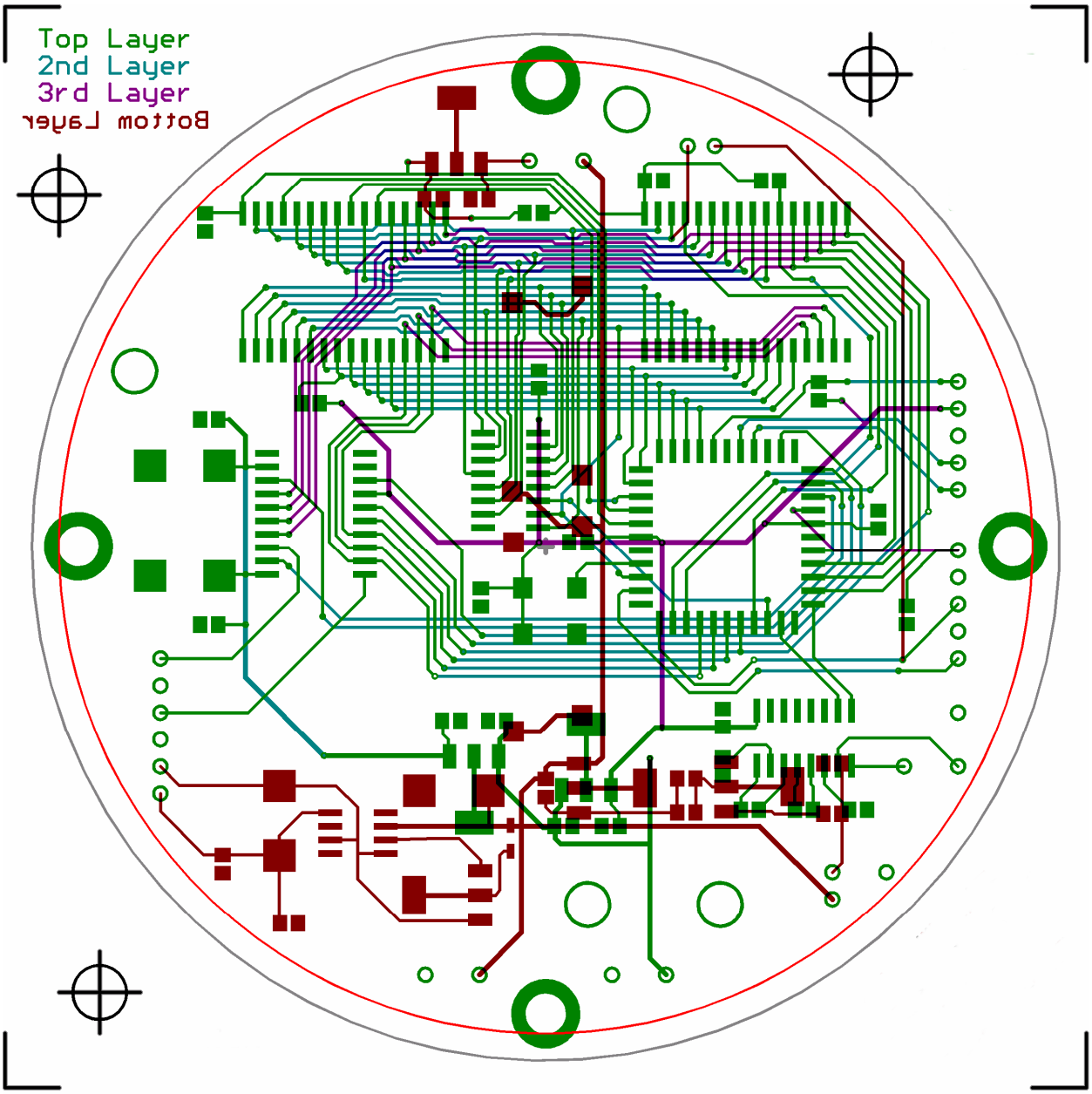

Figure 8. Transparent view of data logger PCB showing connections between all four layers.

Table 3. Procurement information for components used in the ballistic data logger.

Component Description	Schematic Designations	Digi-Key Part No.	Cost/Units
IC MCU FLASH 16KX16 CAN 44PLCC	IC2	PIC18F458-I/L-ND	$185.50/25
IC SOCKET PLCC 44POS SMT	for IC2	ED80010-ND	$15.05/10
IC SRAM 512KX8 LP WIDE 32-SOIC	IC1 and IC4	428-1075-ND	$118.00/25
IC 12STG BINARY COUNTER 16-SOIC	IC3	296-14558-1-ND	$12.00/25
IC ADC 8BIT MPU 2CH MUX 20-SOIC	IC6	ADC08062CIWM-ND	$83.60/25
IC TXRX RS232 1MBPS LP 16-SOIC	IC7	MAX2323CSE-ND	$79.00/25
OSCILLATOR 40MHZ HCMOS SMD	OSC1	CW308-ND	$96.75/10
IC LITH BAT CHRG CTRLR 8-SOIC	BC1	LM3622AM-8.4-ND	$52.73/25
IC VOLT REG FIXED POS SOT-223	VR1, VR2, and VR3	296-12290-1-ND	$31.50/100
CAP .1UF 25V CERAMIC X7R 0805	C1-5, C9-13, C20, C22, and C24	PCC1828CT-ND	$14.20/500
CAP CER .47UF 25V X7R 10% 0805	C14-16, C19, C21, and C23	445-1353-1-ND	$20.80/100
CAP TANTALUM 22UF 16V 20% SMD	C7, C8, C17, and C18	493-2419-1-ND	$80.75/100
RES 1.00 OHM 1/8W 1% 0805 SMD	R2 and R10	311-1.00CCT-ND	$4.93/200
RES 1.00K OHM 1/8W 1% 0805 SMD	R5 and R6	311-1.00KCCT-ND	$4.93/200
RES 4.70K OHM 1/8W 1% 0805 SMD	R4, R11, and R12	311-4.70KCCT-ND	$4.93/200
RES 10.0K OHM 1/8W 1% 0805 SMD	R1, R3, and R7	311-10.0KCCT-ND	$4.93/200
RES 20.0K OHM 1/8W 1% 0805 SMD	R8 and R9	311-20.0KCCT-ND	$4.93/200
TRANS PNP -40V -2000MA SOT-223	T1 and T3	FZT790ACT-ND	$59.13/100
TRANS NPN SW 40V 200MA SOT-223	T2	568-1182-1-ND	$18.90/100
DIODE SWITCH 100V 200MW SOD-323	D1	BAV-19WSDICT-ND	$21.60/100

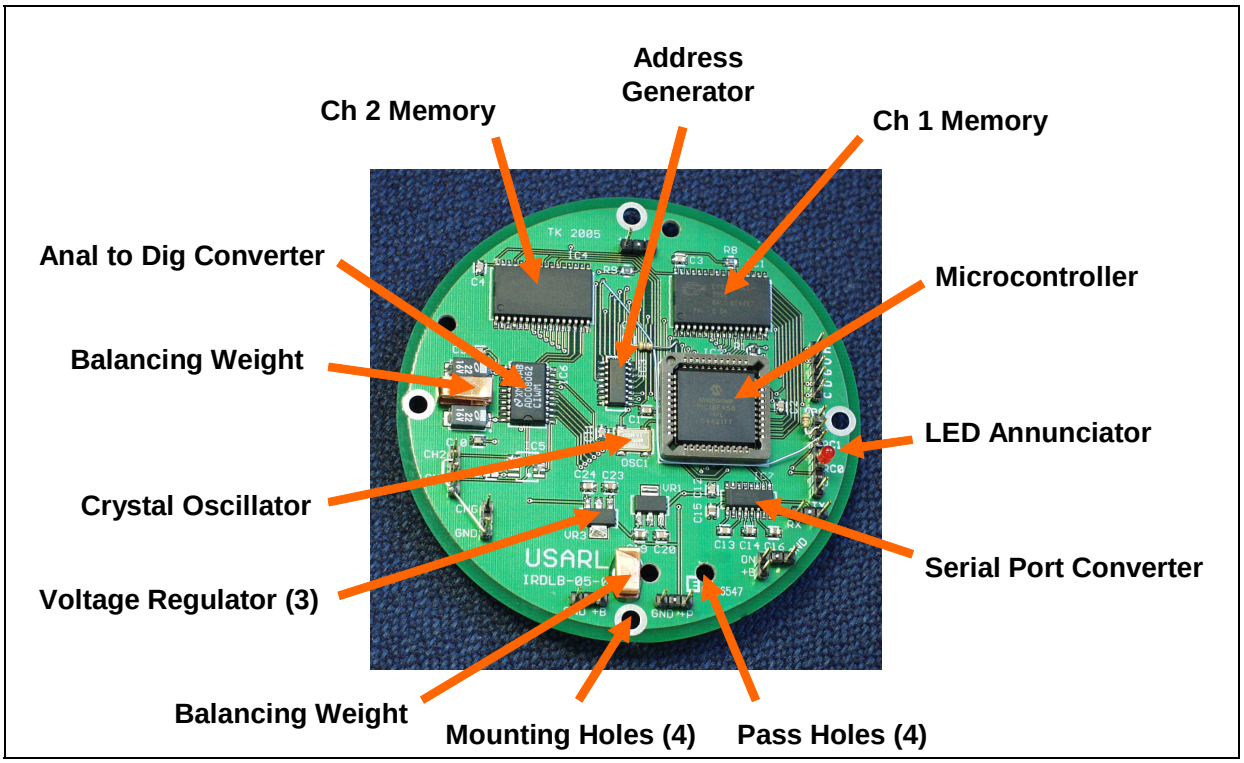

Figure 9. Placement of components on top side of ballistic data logger PCB.

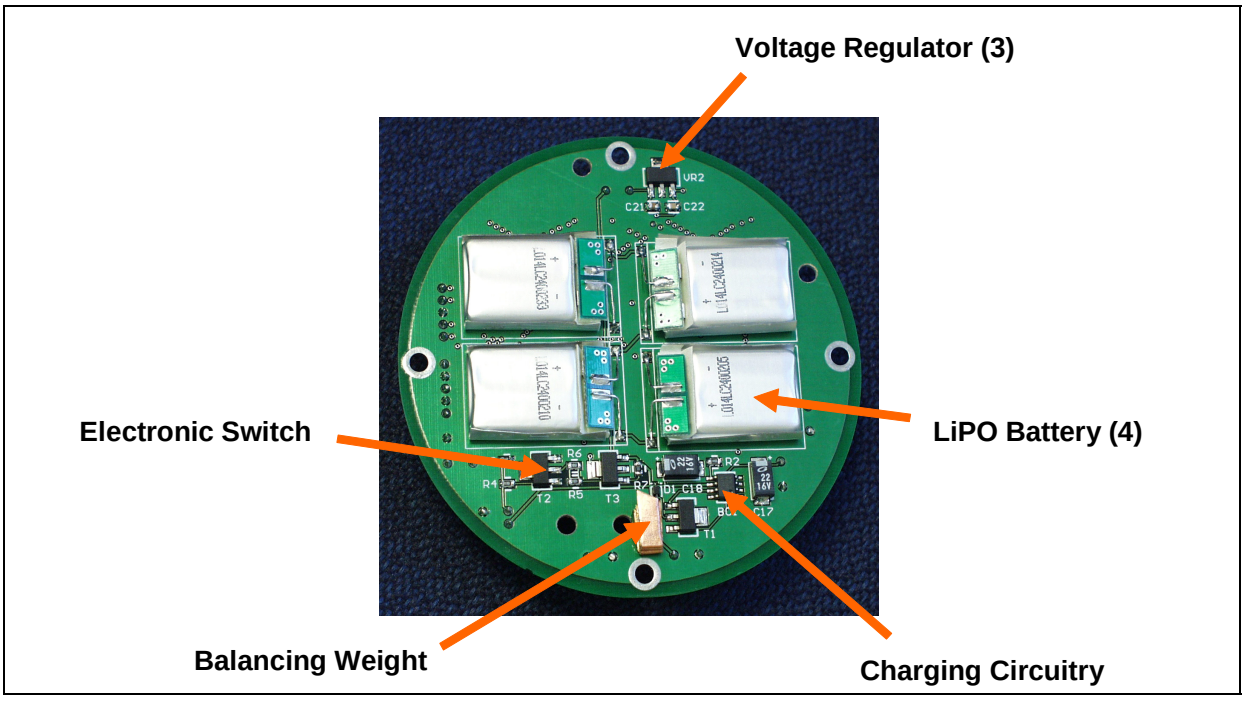

Figure 10. Placement of components on bottom side of ballistic data logger PCB.

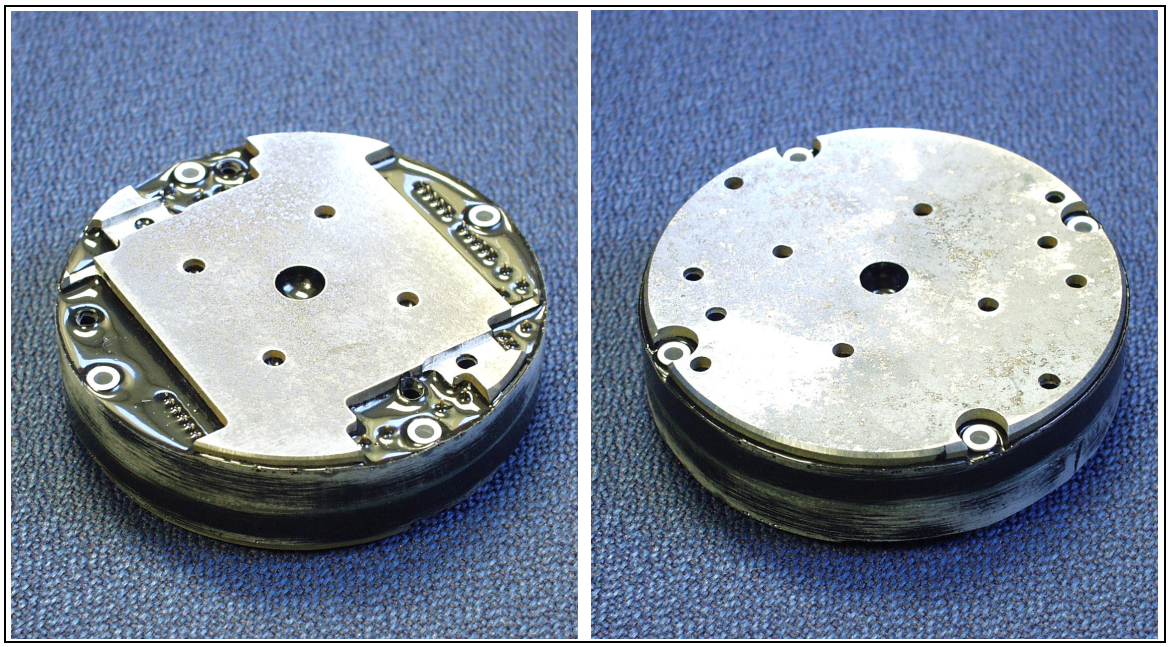

Figure 11. Ballistic data logger in protective packaging.

3. Ballistic Data Logger Software

The ballistic data logger's PIC18F458 microcontroller runs software that is coded in the C programming language and the microcontroller's native assembly language. This mixed-language approach offers access to both the high-level functions provided by C and the inherent speed of assembly language execution. The C *Main* function serves as the software foundation from which additional C and assembly functions are called as required. Microchip's MPLAB C18 compiler (*10*) is combined with the Microchip MPLAB Integrated Development Environment (*11*) to provide the toolset that makes mixed-language programming possible.

The C and assembly codes are presented in appendices A and B along with explicit documentation that often cites specific references where additional information is available. These listings are meant to provide a foundation for individuals who need to recreate and modify the actions of the ballistic data logger.

The microcontroller assembler header file is an ancillary segment of code that maps the microcontroller's special function registers (SFRs) and associated bits to mnemonic identifiers as well as defining the configuration register values. Use of this header file is convenient for a number of reasons. First, equating the SFR register values to mnemonic names streamlines the coding process. For example, it is easier to remember the microcontroller's port A register mnemonic name PORTA than it is to remember its address value of 0xF80. Explicitly defining

the configuration register values in this header file avoids a reliance on configuration register default values that exhibit an unnerving ability to seemingly change at will. Finally, the modular nature of the microcontroller assembler header file facilitates the process of converting the ballistic data logger software to updated hardware that may utilize a different microcontroller. Assuming that the new microcontroller has the same general capabilities and architecture, an appropriately modified header file should allow the legacy code to run on the new hardware. The microcontroller assembler header file is listed in appendix C.

The linker script file defines the memory architecture of the microcontroller so the linker can place code in available ROM memory regions and variables in available RAM memory regions. Regions that are marked PROTECTED are not used for general allocation of program or data. Code or data will only be allocated into these regions if an absolute address is specified for the section or if the section is assigned to the region using a SECTION directive in the linker script file. The linker script file is listed in appendix D.

4. Ballistic Data Logger Test and Evaluation

In this test case the ballistic data logger is used to simultaneously acquire a high-frequency and a low-frequency analog signal to verify the specified data recording time and the maximum data acquisition rate. The acquired data are displayed as a collection of points in figure 12. For display purposes the data record has been divided into quarters with the first recorded data plotted on the left hand side of the top graph and the final recorded data plotted on the right hand side of the bottom graph. Specifically, the low-frequency analog signal is a 10-Hz sine wave that is plotted in red and the high-frequency analog signal is a sine wave with a period of 4.6 μs, which is plotted in blue. Data are acquired at the maximum dual-channel recording rate of 2.3 μs per data pair. Each wavelength of the red 10-Hz sinusoidal curve spans 0.1 s. The fact that there are slightly more than 12 complete wavelengths of the 10-Hz sinusoid substantiates the specified total recording time for the maximum data acquisition rate of 1.21 s, as listed in table 1.

The high-frequency data plotted in blue require additional explanation. With a period of 4.6 μs, the high-frequency sinusoid has a period that is twice the data acquisition time of 2.3 μs per data point. Therefore, successive measurements of the high-frequency sinusoid will be separated in time by half the period and successive measurement values will correspond to sinusoid values that are 180° out of phase. The coarse scale of figure 12 does not allow this behavior to be observed. What is evident is the very long wave beat frequency between the 4.6-μs measurement interval generated by the ballistic data logger and the 4.6-μs sinusoidal period generated by the signal generator that is supplying the analog test signals. Actually, the observance of this beat frequency provides an extremely accurate method for quantifying the data logger's acquisition rate against a known time standard.

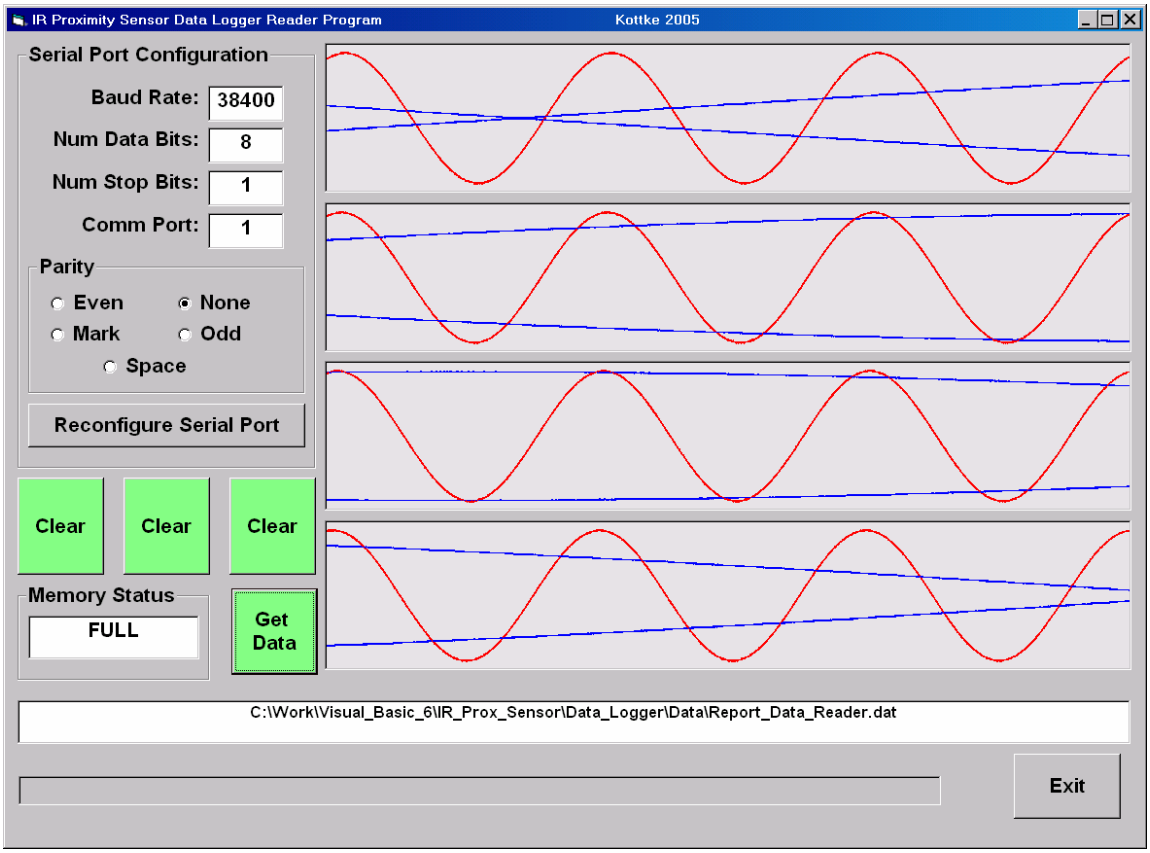

Figure 12. Acquired sinusoidal test data.

The true nature of the high-frequency sinusoidal signal is more evident in figure 13. For this graph, adjacent data points are connected by a line segment. Also, a fifth plot is added at the bottom that magnifies the data region denoted by the two closely spaced vertical cursors on the right hand side of the second plot from the top. Over the short time span of this magnified view, the low-frequency red signal appears constant. However, the high-frequency nature of the blue signal is evident with adjacent data points spanning the entire voltage measurement range. The maximum data acquisition rate of 2.3 µs per channel data pair is therefore verified.

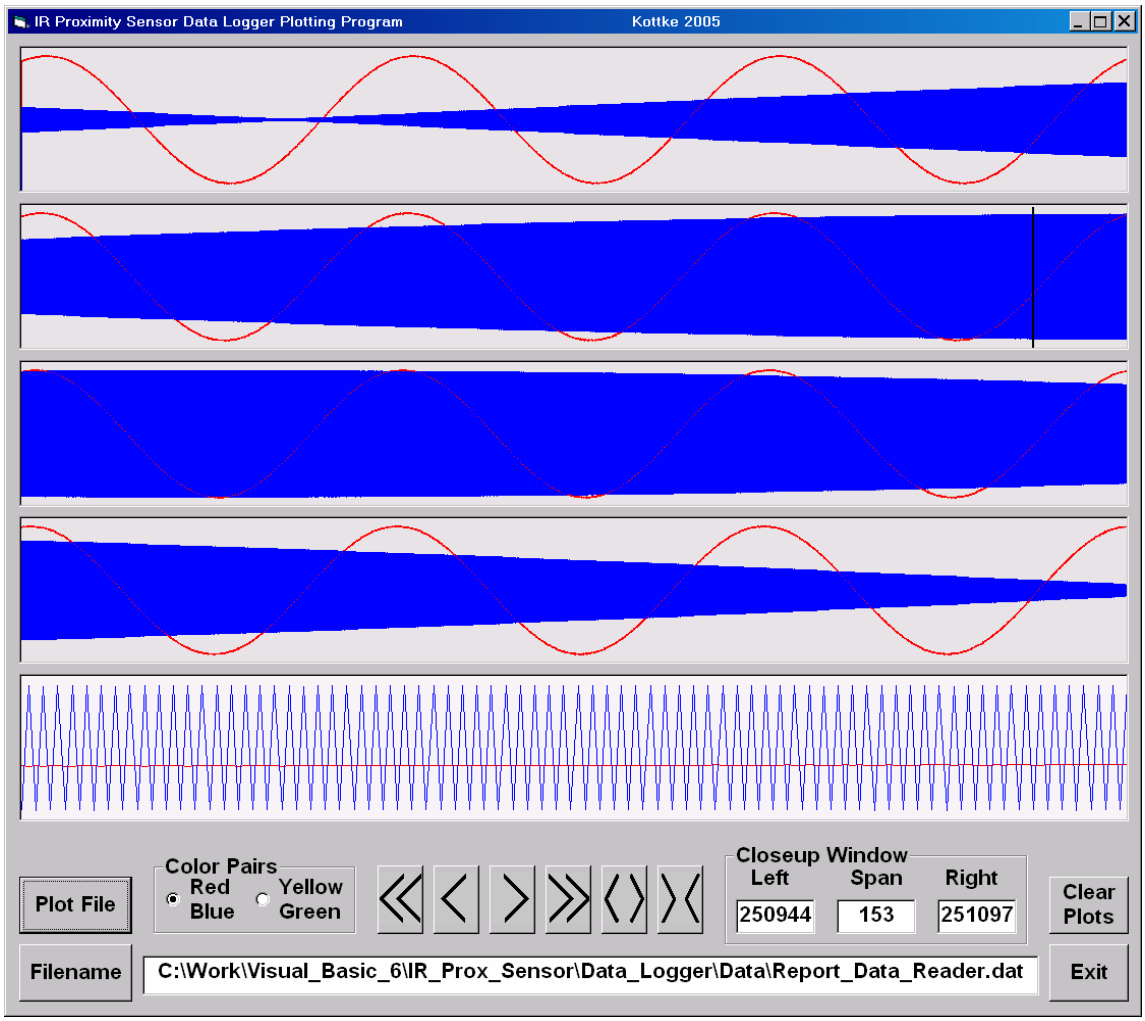

Figure 13. Analysis of acquired sinusoidal test data.

5. References

1. Lithium Polymer Battery Systems Section 3 – Safety, Charging, & Discharging; in the Handbook of FMA, Inc. http://www.fmadirect.com/support_docs/item_1197.pdf (accessed July 2005).
2. National Semiconductor LM3622 Lithium-Ion Battery Charger Controller Datasheet. http://www.national.com/ds/LM/LM3622.pdf (accessed February 2000).
3. Texas Instruments Incorporated. uA78M00 Series Positive-Voltage Regulators Texas Instruments Datasheet. http://focus.ti.com/lit/ds/symlink/ ua78m05/pdf (accessed 2003).
4. Microchip Technology Incorporated. Microchip PIC18FXX8 Data Sheet. http://ww1.microchip.com/downloads/en/devicedoc/41159d.pdf (accessed 2003).
5. National Semiconductor ADC08061/ADC08062 500 ns A/D Converter with S/H Function and Input Multiplexer Datasheet. http://www.national.com/ds/ DC/ADC08061.pdf (accessed June 1999).
6. Cypress Semiconductor Corporation. Cypress CY62148B MoBL 512K x 8 Static RAM Datasheet. http://www.cypress.com/portal/server.pt/ gateway /PTARGS_0_2_1524_209_259_43/http%3B/sjapp20%3B7001/publishedcontent/publish /design_resources/datasheets/contents/cy62148b_5.pdf (accessed October 2001).
7. Texas Instruments Incorporated. CD54HC4040, CD74HC4040, CD54HCT4040, CD74HCT4040 High-Speed CMOS Logic 12-Stage Binary Counter Texas Instruments Datasheet. http://focus.ti.com/lit/ds/symlink/cd74hc4040.pdf (accessed October 2003).
8. Interfacing the Serial/RS232 Port. http://beyondlogic.org/serial/serial.htm (accessed April 2006).
9. 3.0V to 5.5V, Low-Power, up to 1Mbps, True RS-232 Transceivers Using Four 0.1uF External Capacitors Maxim Datasheet. http://pdfserv.maxim-ic.com/en/ds/ MAX3222-MAX3241.pdf (accessed March 1999).
10. Microchip MPLAB C18 Datasheet. http://www.microchip.com/stellent/idcplg?IdcService =SS_GET_PAGE&nodeId=1406&dDocName=en010014 (accessed April 2006).
11. Microchip MPLAB Integrated Development Environment Datasheet. http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&d DocName=en019469 (accessed April 2006).

INTENTIONALLY LEFT BLANK.

Appendix A. Listing of Ballistic Data Logger Microcontroller C Code

This appendix appears in its original form, without editorial change.

```
//<<<<<<------------------ 77 character width template ---------------->>>>>>
/*
                        BALLISTIC DATA LOGGER C CODE

File:                   W2006_04_XXC.c
Author:                 Tom Kottke
Date:                   01 April 2006
Language                MPLAB C18
Microprocessor          PIC18F458
*/

// Specify included files ***************************************************

#include    <p18f458.h>             //microcontroller header file located at
                                    //C:\mcc18\h\p18f458.h
#include    <stdlib.h>              //standard C library header file

// Define constant values ***************************************************

#define     max_mem_size            0x7FFFF     //maximum memory address

// Declare functions ********************************************************

// the following declarations are for the C language functions --------------
            void        main(void);             //main function
            void        low_isr(void);          //low priority interrupt func
            void        high_isr(void);         //hi priority interrupt func
            void        set_low_isr_vector(void);  //set low int rout. vector
            void        set_high_isr_vector(void); //set high int rout vector
            void        tx_string(rom char*);   //transmit a string on USART
            void        tx_START(void);         //send "START" on serial port
            void        tx_mem_all(void);       //dump all mem to serial port
            void        tx_END(void);           //send "END" on serial port
            void        tx_CLEAR(void);         //send "CLEAR on serial port
            void        tx_ERASING(void);       //send "ERASING" on ser. port
            void        tx_FULL(void);          //send "FULL" on serial port
            void        clear_mem(void);        //clear and preset memory
            void        flash_led(char,char,char,char);//blink LED

// the following declarations are for the assembly language functions -------
extern      void        init_interr(void);      //initialize the interrupts
extern      void        init_PORT_A(void);      //initialize port A
extern      void        init_PORT_B(void);      //initialize port B
extern      void        init_PORT_C(void);      //initialize port C
extern      void        init_PORT_D(void);      //initialize port D
extern      void        init_PORT_E(void);      //initialize port E
extern      void        init_serial(void);      //initialize serial port
extern      void        init_ADC(void);         //initialize anal to dig conv
extern      void        set_interr(void);       //reconfigure interrupts
extern      void        rst_mem_add(void);      //reset hardware mem. address
extern      void        tx_char(char);          //transmit character on USART
extern      char        read_mem_0(void);       //read mem location in chip 0
```

```
extern      char        read_mem_1(void);       //read mem location in chip 1
extern      void        inc_mem_add(void);      //increment memory address
extern      char        rx_poll(void);          //determine if rx occurred
extern      char        rx_char(void);          //capture serial rx byte
extern      void        write_mem_0(char);      //write to mem loc. in chip 0
extern      void        write_mem_1(char);      //write to mem loc. in chip 1
extern      char        test_mem_clear(void);   //test for clear memory
extern      void        pre_conf_acq(void);     //configure for data acquis
extern      void        get_data_0(void);       //acquire channel 0 data
extern      void        get_data_1(void);       //acquire channel 1 data
extern      void        post_conf_acq(void);    //post data acq configuration
extern      void        time_delay(char,char,char); //generate time delay
extern      void        turn_led_on(void);      //turn on annunciator LED
extern      void        turn_led_off(void);     //turn off annunciator LED
extern      void        clear_int_flag(void);   //clears INT0 interrupt flag
extern      void        acquire_data(void);     //acquires all data
extern      char        test_interrupt(void);   //tests interrupt value

// Set up high priority interrupt jump vector ******************************

#pragma code hi_int_vector                      //hi_int_vector refers to a
void        set_high_isr_vector(void)           //section of ROM program
{                                               //memory space as defined in
            _asm                                //the linker script from
            GOTO        high_isr                //0x0008 to 0x0017. high_isr
            _endasm                             //refers to the location of
}                                               //a C language function.
                        //_asm and _endasm delimit in-line assembly operation

// Main code ***************************************************************

#pragma code main_vector            //when compiled, the following code will
                                    //be located in the ROM program memory
                                    //space "main_vector" as defined in the
                                    //linker script starting at 0x0800

void        main(void)
{
            // the following code initializes the microcontroller on power-up

            time_delay(16,255,255); // call assembly language routine to
                                    // generate power-up time delay
            init_interr();          // call assembly language routine to
                                    // turn off interrupts
            init_PORT_A();          // call assembly language routine to
                                    // initialize port A
            init_PORT_B();          // call assembly language routine to
                                    // initialize port B
            init_PORT_C();          // call assembly language routine to
                                    // initialize port C
            init_PORT_D();          // call assembly language routine to
                                    // initialize port D
            init_PORT_E();          // call assembly language routine to
                                    // initialize port E
            init_serial();          // call assembly language routine to
                                    // initialize serial port
            init_ADC();             // call assembly language routine to
```

```
                                    // initialize analog to digital converter
            set_interr();           // call assembly language routine to
                                    // turn on desired interrupts
            flash_led(1,19,4,106);  // visually display power on status
                                    // by flashing LED once

}
//_____________________________________________________________________________
void  tx_FULL(void)     //routine to output the word FULL
{
      rom near static unsigned char *string_tx; //pointer to ROM
                              //text data

      string_tx = "FULL";     //keyword output to signal that the
      tx_string(string_tx);   // memory is full
}
//_____________________________________________________________________________
void  clear_mem(void)   //routine to clear and preset memory
{
      near long   l;          //declare long integer
      near char   index=0;          //declare char

            rst_mem_add();                      //reset mem add
            if(test_mem_clear())    // visually display clear memory status
            {
                        flash_led(9,19,4,106);  //flash LED 9 times if
            }                                   //memory is clear

// the following loop is the default code that is run continuously after the
// microcontroller is initialized and when data is not being acquired. this
// loop outputs the contents of the memory through the serial port.

loop:
            if(test_mem_clear())    //if mem is clear
            {
                        tx_CLEAR(); //then send the word "CLEAR" on ser. port
            }
            else
            {
                        tx_FULL();  //else send the word "FULL" on ser. port
            }
            tx_START();             //send word "START" on serial port
            tx_mem_all();           //dump entire memory to serial port
            tx_END();               //send word "END" on serial port

            if(rx_poll())                       //if ser port has rec data...
            {
                        if(rx_char()==0b01010101)//if serial port value = 85
                        {
                                    tx_ERASING();//send ERASING on ser. port
                                    clear_mem(); //clear and preset memory
                                    tx_CLEAR();  //send CLEAR on ser. port
                        }
            }
            goto loop;
}
```

```
// C Functions Code Listing ************************************************

void        tx_string(rom near unsigned char *tran_string)//routine to output
{                                           //a string along with the
                                            //end-of-string delimiter, 0
            near unsigned char j=0;         //declare char in access mem

            while(*(tran_string+j))
            {
                        tx_char(*(tran_string + j++));      //call to assem.

                                                    //language rout.
            }
            tx_char(*(tran_string + j));        //output end-of-string marker
}

//_____________________________________________________________________________
void        tx_mem_all(void)
{
            near long   l;          //declare long in access data memory
            near int i;             //declare integer in access data memory

            rst_mem_add();          //call assembly language routine
                                    // to reset memory address to zero
            for (l=0x00 ; l<=max_mem_size ; ++l)
            {
                        init_PORT_D();          //call assembly routine
                                                //to conf port D for input
                        tx_char(read_mem_0());  //call assembly routine
                                                //to output data value on ser
                        tx_char(read_mem_1());  //call assembly routine
                                                //to output data value on ser
                        inc_mem_add();          //call assembly routine
                                                //to output carriage return
            }
            if (test_mem_clear())   //display memory status on LED
            {
                        flash_led(10,19,4,106); //flash led 10 times
            }
            else
            {
                        flash_led(1,19,4,106);  //flash led once
            }
}
//_____________________________________________________________________________

void        tx_START(void)          //routine to output the word START
{
            rom near static unsigned char *string_tx; //pointer to ROM
                                                      //text data

            string_tx = "START";    //keyword output to signal start of
            tx_string(string_tx);   // computer memory dump
}
//_____________________________________________________________________________
void        tx_END(void)            //routine to output the word END
{
```

```
            rom near static unsigned char *string_tx; //pointer to ROM
                                                      //text data

            string_tx = "END";      //keyword output to signal end of
            tx_string(string_tx);   // computer memory dump
}
//_____________________________________________________________________________
void        tx_CLEAR(void)          //routine to output the word CLEAR
{
            rom near static unsigned char       *string_tx; //pointer to ROM
                                                      //text data

            string_tx = "CLEAR";    //keyword output to signal clear and
            tx_string(string_tx);   // preset memory
}
//_____________________________________________________________________________
void        tx_ERASING(void)        //routine to output the word ERASING
{
            rom near static unsigned char       *string_tx; //pointer to ROM
                                                      //text data

            string_tx = "ERASING";  //keyword output to signal memory is
            tx_string(string_tx);   // being erased


            for( l=0x00 ; l<=max_mem_size ; ++l,++index)//cycling through
            {                                   // all memory addresses
                        write_mem_0(index);     //loading them with ramp
                        write_mem_1(index);     //function values
                        inc_mem_add();          //increment address value
            }
}
//_____________________________________________________________________________
void        flash_led(char N,char D3,char D2,char D1) //routine to
                                                //flash LED specified number
                                                //of times at specified rate
{
            near unsigned char      i;          //declare char in access mem
            for( i=0 ; i<N ; ++i)
            {
                        turn_led_on();          //ass. call to turn LED on
                        time_delay(D3,D2,D1);   //ass. call to gen time delay
                        turn_led_off();         //ass. call to turn LED off
                        time_delay(D3,D2,D1);   //ass. call to gen time delay
            }
}

// Interrupt Service Routine Codes ****************************************

#pragma interruptlow low_isr
void        low_isr(void)
{
//          not utilized at this time
}
//_____________________________________________________________________________

#pragma interrupt high_isr          //interrupt code to acquire data
```

```
                                    //high priority interrupt is accessed by
void        high_isr(void)          //high level on RB0/INT0, pin 36
{
            near int i;             //declare integer i in access data memory

            for (i=0 ; i<100 ; ++i)             //testing the validity of the
            {                                   //interrupt signal by
                        if(!test_interrupt())   //sampling it multiple times
                        {
                                    goto false;
                        }
            }

            if(test_mem_clear())                //if the memory is clear
            {
                        rst_mem_add();          //reset memory address
                        pre_conf_acq();         //configure for data acquis

                        acquire_data();
                        post_conf_acq();        //conf for post data acquis
                        flash_led(3,19,4,106);  //flash LED 3 times
                        clear_int_flag();       //clear the interrupt flag
            }
            else
            {
                        clear_int_flag();       //clear the interrupt flag
            }
false:      clear_int_flag();                   //clear the interrupt flag

}
```

INTENTIONALLY LEFT BLANK.

Appendix B. Listing of Ballistic Data Logger Microcontroller Assembly Code

Thie appendix appears in its original form, without editorial change.

```
;<<<<<<---------------------- 77 character width template -------------->>>>>>
;
;                       BALLISTIC DATA LOGGER ASSEMBLY CODE
;
;File:                  W2006_04_xxa.asm
;Author:                Tom Kottke
;Date:                  01 April 2006
;Language               MPLAB Assember
;Microprocessor         PIC18F458

; Specify included files ***************************************************

#include    <TK_PIC18F458_DL.inc>   ;TK specific header file located at
                                    ;C:\Work\PIC\IR_Prox\D_Log\W2005_07\...


; Declare functions ********************************************************


            global      init_interr             ;declaring assembly language
            global      init_PORT_A             ;routines as global so they
            global      init_PORT_B             ;can be called by C code
            global      init_PORT_C
            global      init_PORT_D
            global      init_PORT_E
            global      init_serial
            global      init_ADC
            global      set_interr
            global      rst_mem_add
            global      tx_char
            global      read_mem_0
            global      read_mem_1
            global      inc_mem_add
            global      rx_poll
            global      rx_char
            global      write_mem_0
            global      write_mem_1
            global      test_mem_clear
            global      pre_conf_acq
            global      get_data_0
            global      get_data_1
            global      post_conf_acq
            global      time_delay
            global      turn_led_on
            global      turn_led_off
            global      clear_int_flag
            global      acquire_data
            global      test_interrupt

; Reserve space and declare names of uninitialized variables in access RAM **

            udata_acs                           ;directive declares the
```

```
                                        ;beginning of a section of
                                        ;access uninitialized data.
i           res         1               ;syntax is variable name, key
delay_cnt_1 res         1               ;word res, number of bytes to
delay_cnt_2 res         1               ;be reserved
delay_cnt_3 res         1
count_1     res         1
count_2     res         1

; Code section ********************************************************

            code                        ;directive declares the
                                        ;beginning of a section of
                                        ;program code

;ASSEMBLY FUNCTIONS TO BE CALLED BY C LANGUAGE PROGRAM **********************
;_____________________________________________________________________________
;Name:      init_interr
;Purpose:   To initialize the PIC18F458 interrupts when the microcontroller
;           is first powered up to benign settings that will not interfere
;           with subsequent initialization activities
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Interrupt priority levels are enabled, high priority interrupts
;           are disabled, low priority interrupts are disabled, timer 0
;           overflow interrupt is disabled, INT0 external interrupt is dis-
;           abled, port B change interrupts are disabled, all port B pull-
;           ups are disabled
;Notes:     Registers RCON, INTCON, and INTCON2 are special function
;           registers that are accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 58, 79, 80, and 47

init_interr
            BSF         RCON,IPEN,0             ;enable priority levels on
                                                ; interrupts
            BCF         INTCON,GIE_GIEH,0       ;disable all high prior inter
            BCF         INTCON,PEIE_GIEL,0      ;disable all low prior interr
            BCF         INTCON,TMR0IE,0         ;disable T0 overflow interr
            BCF         INTCON,INT0IE,0         ;disable INT0 ext interr
            BCF         INTCON,RBIE,0           ;disable PORT B change interr
            BSF         INTCON2,RBPU_,0         ;disable PORT B pull ups

            RETURN
;_____________________________________________________________________________
;Name:      init_PORT_A
;Purpose:   To configure the 7-bit wide bi-directional port A of the
;           PIC18F458 microcontroller for digital I/O.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Port A output data latches are cleared, port A is configured for
;           digital I/O on all seven pins, and all 7 bits are configured as
;           outputs.
;Notes:     This port is used to set the seven least significant bits of the
;           memory address. The highest bit of this port also supplied the
;           clock pulse for the 74HC4040 that supplies the remaining address
;           bits. Registers LATA, ADCON1, and TRISA are special function
;           registers that are accessed through the access bank.
```

```
;References:PIC18FXX8 Data Sheet pages 93, 242 and 47

init_PORT_A
            CLRF        LATA,0                  ;clear output data latches
            MOVLW       07h                     ;configure PORT A for digital
            MOVWF       ADCON1,0                ; I/O on all pins
            MOVLW       b'00000000'             ;PORT A I/O direct. template
            MOVWF       TRISA,0                 ;define PORT A I/O directions

            RETURN
;_____________________________________________________________________________
;Name:      init_PORT_B
;Purpose:   To configure the 8-bit wide bi-directional port B of the
;           PIC18F458 microcontroller for digital I/O.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Port B output data latches are cleared, bits 0 and 5-7 are
;           configured as inputs while bits 1 through 4 are configured as
;           outputs.
;Notes:     Bit 0 of port B is configured as an input and is available as an
;           interrupt source. Bits 1 through 4 are used to control the
;           operation of the ADC08062 analog to digital conversion chip.
;           Specifically, bit 1 determines which of the two ADC inputs is
;           active, bit 2 controls the active-low chip select, bit 3 controls
;           the active-low read line, and bit 4 controls the active-low write
;           line. Bit 5 is not utilized and bits 6 and 7 are reserved for In-
;           Circuit-Serial-Programming operations.Registers LATB and TRISB
;           are special function registers that are accessed through the
;           access bank.
;References:PIC18FXX8 Data Sheet pages 96 and 47

init_PORT_B
            CLRF        LATB,0                  ;clear output data latch
            MOVLW       b'00011100'             ;port B initial latch values
            MOVWF       LATB,0                  ;set port B latch values
            MOVLW       b'11100001'             ;PORT B I/O direct. template
            MOVWF       TRISB,0                 ;define PORT B I/O directions

            RETURN
;_____________________________________________________________________________
;Name:      init_PORT_C
;Purpose:   To configure the 8-bit wide bi-directional port C of the
;           PIC18F458 microcontroller for digital I/O.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Port C output data latches are cleared, bits 0 through 6 are
;           configured as outputs and bit 7 is configured as an input.
;Notes:     Bit 0 is configured as an output and is available to act as an
;           event annunciators. Bit 1 is configured as an input to monitor
;           the most significant bit of the address bus so that data
;           acquisition can be terminated at the appropriate time. Bits 2
;           through 4 are outputs that control  the actions of memory chip 0.
;           Specifically, bit 2 controls the active-low chip enable line, bit
;           3 controls the active-low write     enable line, and bit 4
;           controls the active-low output enable line. Bit 5 is an output
;           that provides an active-high reset signal to the 74HC4040 memory
;           address counter counter. Bits 6 and 7 are configured at an output
```

```
;           and input respectively and are reserved for use by the serial
;           port. Registers LATC and TRISC are  special function registers
;           that are accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 100 and 47

init_PORT_C
            CLRF        LATC,0                  ;clear output data latch
            MOVLW       b'00011100'             ;port C initial latch values
            MOVWF       LATC,0                  ;set port C latch values
            MOVLW       b'10000010'             ;PORT C I/O direct. template
            MOVWF       TRISC,0                 ;define PORT C I/O directions

            RETURN
;_____________________________________________________________________________
;Name:      init_PORT_D
;Purpose:   To configure the 8-bit wide bi-directional port D of the
;           PIC18F458 microcontroller for digital input.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Port D output data latches are cleared, comparator functions are
;           disabled, and all eight bits are configured as inputs.
;Notes:     All eight bits of port D are used to access the 8-bit data bus.
;           These bits are initially set as high impedence inputs that will
;           not interefer with the data acquisition process. Registers LATD,
;           CMCON, and TRISD are special function registers that are accessed
;           through the access bank.
;References:PIC18FXX8 Data Sheet pages 102, 249 and 47

init_PORT_D
            CLRF        LATD,0                  ;clear output data latch
            MOVLW       b'00000111'             ;disable all comparator
            MOVWF       CMCON,0                 ; functions
            MOVLW       b'11111111'             ;PORT D I/O direct. template
            MOVWF       TRISD,0                 ;define PORT D I/O directions

            RETURN
;_____________________________________________________________________________
;Name:      init_PORT_E
;Purpose:   To configure the 3-bit wide bi-directional port E of the
;           PIC18F458 microcontroller for digital output.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Port E output data latches are cleared, and all three bits are
;           configured as outputs.
;Notes:     The three bits of port E are used to control memory chip 1.
;           Specifically, bit 0 controls the active-low chip enable line, bit
;           1 controls the active-low write enable line, and bit 2 controls
;           the active-low output enable line. Registers LATE, and TRISE are
;           special function registers that are accessed through the access
;           bank.
;References:PIC18FXX8 Data Sheet pages 104 and 47

init_PORT_E
            CLRF        LATE,0                  ;clear output data latch
            MOVLW       b'111'                  ;port E initial latch values
            MOVWF       LATE,0                  ;set port E latch values
            MOVLW       b'000'                  ;PORT E I/O direct. template
```

```
            MOVWF       TRISE,0                 ;define PORT E I/O directions

            RETURN
;_____________________________________________________________________________
;Name:      init_serial
;Purpose:   To configure the Universal Synchronous/Asynchronous Receiver
;           Transmitter (USART) on the PIC18F458 microcontroller for serial
;           communications with an external monitor/server personal computer.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   The microcontroller serial port is configured for 8-bit trans-
;           mission, serial transmission is enabled, the USART is configured
;           for asynchronous communication, the high speed baud rate genera-
;           tion mode is selected, the serial port is enabled, 8-bit
;           reception is configured, continuous serial port reception is
;           enabled, address detection is disabled, the serial port baud rate
;           generator is configured for a baud rate of 38400, the USART
;           receive interrupt is set to high priority, and the USART receive
;           interrupt is enabled.
;Notes:     The PIC18F458 microcontroller uses bits 6 and 7 of port C to
;           transmit and receive serial data. Registers TXSTA, RCSTA, SPBRG,
;           IPR1, and PIE1 are special function registers that are accessed
;           through the access bank.
;References:PIC18FXX8 Data Sheet pages 183-192, 88, 85, and 47

init_serial
            MOVLW       b'00100110'             ;transmit status reg template
            MOVWF       TXSTA,0                 ;define transmit status reg
            MOVLW       b'10010000'             ;receive status reg template
            MOVWF       RCSTA,0                 ;define receive status regist
            MOVLW       d'64'                   ;baud rate gener 38.4 Kbps
            MOVWF       SPBRG,0                 ;define baud rate gener reg
            BSF         IPR1,RCIP,0             ;set rx interrupt to hi prior
            BCF         PIE1,RCIE,0             ;disable USART rx interrupt

            RETURN
;_____________________________________________________________________________
;Name:      init_ADC
;Purpose:   To initialize the ADC08062 analog to digital converter to benign
;           settings that will not interfere with subsequent activities.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   The ADC08062 active-low chip select line is pulled high, the
;           active-low write line is pulled high, and the active-low read
;           line is pulled high.
;Notes:     The chip select line of the ADC08062 analog to digital converter
;           is controlled by bit 2 of port B which has been equated in the
;           included file TK_PIC18F458_DL.inc to "CS". Similarily, the write
;           line is controlled by bit 4 which has been equated to "WR" and
;           the read line is controlled by bit 3 which has been equated to
;           "RD". Register PORTB is a special function register that is
;           accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 96 and 47

init_ADC
            BSF         PORTB,CS,0              ;set active-lo chip select
```

```
            BSF         PORTB,WR,0              ;set active-lo write line
            BSF         PORTB,RD,0              ;set active-lo read line

            RETURN
;_____________________________________________________________________________
;Name:      set_interr
;Purpose:   To reconfigure the PIC18F458 interrupts after the microcontroller
;           has been initialized.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Interrupt 0 is configured to activate on a rising edge at RB0,
;           INT0 is enabled, and all high priority interrupts are enabled.
;Notes:     INT0 is always a high priority interrupt source. Registers
;           INTCON2 and INTCON are special function registers that are
;           accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 80, 79, and 47

set_interr
            BSF         INTCON2,INTEDG0,0       ;interr on INT0 rising edge
            BSF         INTCON,INT0IE,0         ;enable INT0 ext interr
            BSF         INTCON,GIE_GIEH,0       ;enable all high prior inter

            RETURN
;_____________________________________________________________________________
;Name:      rst_mem_add
;Purpose:   To reset the memory address bus.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   The latch of port A, which provides the 7 least significant bits
;           of the memory address, is cleared and a positive pulse is
;           supplied to the memory reset line of the 74HC4040 to zero the
;           remaining 12 bits.
;Notes:     The master reset line of the 74HC4040 is controlled by bit 5 of
;           port C which has been equated in the included file
;           TK_PIC18F458_DL.inc to "MR". Registers LATA and PORTC are special
;           function registers that are accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 100 and 47.

rst_mem_add
            CLRF        LATA,0                  ;clear port A output
            BSF         PORTC,MR,0              ;generate a short positive
            NOP                                 ;pulse on the 74HC4040 master
            BCF         PORTC,MR,0              ;reset line

            RETURN
;_____________________________________________________________________________
;Name:      tx_char
;Purpose:   To transmit a single ASCII character through the serial port on
;           the PIC18F458 microcontroller.
;Passed:    A 1-byte ASCII character value is passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   The transmission buffer is determined to be empty, the passed
;           argument is recovered from the software stack and transmitted
;           through the serial port.
;Notes:     This routine waits for any previous serial transmissions to be
;           completed by monitoring the TXIF bit of the peripheral interrupt
;           request register, PIR1, which remains low until the transmit
```

```
;           buffer is empty. When this assembly routine is called by the C
;           program the single argument is pushed onto the top of the
;           software stack and the stack pointer is incremented to the next
;           available empty stack location. The argument value is recovered
;           by accessing the memory location of the current stack pointer
;           decremented by one. This is accomplished by first setting the
;           working register WREG to a value of minus one (0xFF) and using
;           the indirect addressing operation PLUSW1 that uses WREG as an
;           offset to the current stack pointer.  Register PIR1 is a special
;           function register that is accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 55, 82, 189, and 47. MPLAB C18 C
;           Compiler User's Guide pages 38 - 45.
tx_char
tx_clear1   BTFSS       PIR1,TXIF,0             ;is TXREG empty ?
            GOTO        tx_clear1               ;NO, retest TXREG status

            MOVLW       h'FF'                   ;load -1 into WREG
            MOVFF       PLUSW1,TXREG            ;move FSR1 offset by WREG
            NOP                                 ;delay to accomidate delay in
            NOP                                 ; updating of TXIF

            RETURN
;_____________________________________________________________________________
;Name:      read_mem_0
;Purpose:   To read and return a single 8-bit value from the addressed
;           memory location in memory chip 0.
;Passed:    No variables are passed to this routine.
;Returned:  One 8-bit value is returned from this routine.
;Actions:   The active-low write enable line is forced high to enable a
;           memory location read, the active-low chip enable line is forced
;           low to activate the chip, the active-low output enable line is
;           forced low to present the memory data to the data bus, the data
;           on the data bus is input through port D of the microcontroller
;           and transferred to the WREG register, the output enable line is
;           forced high to clear the data bus, and the chip enable line is
;           forced high to place the memory chip in standby mode.
;Notes:     8-bit arguments are transferred from assembly functions to C
;           programs in the WREG register. The write enable line of the
;           memory chip 0 is controlled by bit 3 of port C which has been
;           equated in the included file TK_PIC18F458_DL.inc to "WE0".
;           Similarily, the chip enable line is controlled by bit 2 which has
;           been equated to "CE0" and the output enable line is controlled by
;           bit 4 which has been equated to "OE0". Registers PORTC and PORTD
;           are special function registers that are accessed through the
;           access bank.
;References:PIC18FXX8 Data Sheet pages 100, 102 and 47. MPLAB C18 C Compiler
;           User's Guide pages 37.
read_mem_0
            BSF         PORTC,WE0,0             ;set active-lo write enable
            BCF         PORTC,CE0,0             ;clear active-lo chip enable
            BCF         PORTC,OE0,0             ;clear active-lo output enab
            MOVF        PORTD,0,0               ;move data at port D to WREG
            BSF         PORTC,OE0,0             ;set active_lo output enable
            BSF         PORTC,CE0,0             ;set active_lo chip enable

            RETURN
;_____________________________________________________________________________
```

```
;Name:      read_mem_1
;Purpose:   To read and return a single 8-bit value from the addressed
;           memory location in memory chip 1.
;Passed:    No variables are passed to this routine.
;Returned:  One 8-bit value is returned from this routine.
;Actions:   The active-low write enable line is forced high to enable a
;           memory location read, the active-low chip enable line is forced
;           low to activate the chip, the active-low output enable line is
;           forced low to present the memory data to the data bus, the data
;           on the data bus is input through port D of the microcontroller
;           and transferred to the WREG register, the output enable line is
;           forced high to clear the data bus, and the chip enable line is
;           forced high to place the memory chip in standby mode.
;Notes:     8-bit arguments are transferred from assembly functions to C
;           programs in the WREG register. The write enable line of the
;           memory chip 1 is controlled by bit 1 of port E which has been
;           equated in the included file TK_PIC18F458_DL.inc to "WE1".
;           Similarily, the chip enable line is controlled by bit 0 which has
;           been equated to "CE1" and the output enable line is controlled by
;           bit 2 which has been equated to "OE1". Registers PORTD and PORTE
;           are special function registers that are accessed through the
;           access bank.
;References:PIC18FXX8 Data Sheet pages 102, 104 and 47. MPLAB C18 C Compiler
;           User's Guide pages 37.
read_mem_1
            BSF         PORTE,WE1,0             ;set active-lo write enable
            BCF         PORTE,CE1,0             ;clear active-lo chip enable
            BCF         PORTE,OE1,0             ;clear active-lo output enab
            MOVF        PORTD,0,0               ;move data at port D to WREG
            BSF         PORTE,OE1,0             ;set active_lo output enable
            BSF         PORTE,CE1,0             ;set active_lo chip enable

            RETURN
;_____________________________________________________________________________
;Name:      inc_mem_add
;Purpose:   To increment the memory address value.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Port A, which provides the 7 least significant bits
;           of the memory address, is incremented. If this causes port A to
;           roll over from 0xFF to 0x00, then the most significant bit of
;           port A, which is connected to the clock pulse line of the
;           74HC4040, will cause the 74HC4040 to increment its value as well.
;Notes:     Register port A is a special function registers that is accessed
;           through the access bank.
;References:PIC18FXX8 Data Sheet pages 93 and 47.

inc_mem_add
            INCF        PORTA,1,0               ;increment port A (LSBs)


            RETURN
;_____________________________________________________________________________
;Name:      rx_poll
;Purpose:   To return a non-zero value if data has been received by the USART
;           serial port and a value of 0 if data has not been received.
;Passed:    No variables are passed to this routine.
```

```
;Returned:  One 8-bit value is returned from this routine.
;Actions:   The working register WREG is set if data has been received by the
;           serial port or WREG is cleared if data has not been received.
;Notes:     The reception of serial port data is denoted by the receive
;           interrupt flag, RCIF, of the peripheral interrupt request
;           register number one, PIR1, being set. Conversely, if RCIF is
;           clear, this denotes that serial port data has not been received.
;           8-bit arguments are transferred from assembly functions to C
;           programs in the WREG register. Register PIR1 is a special
;           function registers that is accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 82 and 47. MPLAB C18 C Compiler User's
;           Guide pages 37.
rx_poll
            BTFSS       PIR1,RCIF,0             ;has ser port received data?
            CLRF        WREG,0                  ;NO, clear WREG
            BTFSC       PIR1,RCIF,0             ;has ser port received data?
            SETF        WREG,0                  ;YES, set WREG

            RETURN
;_____________________________________________________________________________
;Name:      rx_char
;Purpose:   To return a single ASCII character that has been received by the
;           USART serial port.
;Passed:    No variables are passed to this routine.
;Returned:  One 8-bit value is returned from this routine.
;Actions:   The contents of the serial port receive register, RCREC, are
;           moved to the working register, WREG, and the serial port receive
;           interrupt flag bit, RCIF, of the peripheral interrupt request
;           register number one, PIR1, is cleared.
;Notes:     8-bit arguments are transferred from assembly functions to C
;           programs in the WREG register. Registers PIR1 and RCREG are
;           special function registers that are accessed through the access
;           bank.
;References:PIC18FXX8 Data Sheet pages 191, 82 and 47. MPLAB C18 C Compiler
;           User's Guide pages 37.
rx_char
            MOVF        RCREG,0,0               ;move received data to WREG
            BCF         PIR1,RCIF,0             ;clear serial port rx flag

            RETURN
;_____________________________________________________________________________
;Name:      write_mem_0
;Purpose:   To write the single byte of data that is passed to this routine
;           to the addressed memory location in chip 0.
;Passed:    A 1-byte value is passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Port D is configured as an output port, memory chip 0 is
;           configured for input and activated, the byte value to be written
;           is recovered from the stack and output onto the data bus through
;           port D, the data value is captured into memory chip 0 by pulsing
;           the active-low write enable line low, the memory chip is placed
;           back into power-down mode, and port D is returned to input port
;           status.
;Notes:     When this assembly routine is called by the C program the single
;           argument is pushed onto the top of the software stack and the
;           stack pointer is incremented to the next available empty stack
;           location. The argument value is recovered by accessing the memory
```

```
;           location of the current stack pointer decremented by one. This
;           is accomplished by first setting the working register WREG to a
;           value of minus one (0xFF) and using the indirect addressing
;           operation PLUSW1 that uses WREG as an offset to the current stack
;           pointer. The write enable line of the memory chip 0 is controlled
;           by bit 3 of port C which has been equated in the included file
;           TK_PIC18F458_DL.inc to "WE0". Similarily, the chip enable line is
;           controlled by bit 2 which has been equated to "CE0" and the
;           output enable line is controlled by bit 4 which has been equated
;           to "OE0". Registers TRISD, PORTC, and PORTD are special function
;           registers that are accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 102, 100, and 47. MPLAB C18 C Compiler
;           User's Guide pages 38 - 45.
write_mem_0
            MOVLW       b'00000000'             ;configure port D for output
            MOVWF       TRISD,0                 ; by modifying direc template
            BSF         PORTC,OE0,0             ;set active-lo output enable
            BCF         PORTC,CE0,0             ;clear active-lo chip enable
            MOVLW       h'FF'                   ;load -1 into WREG
            MOVFF       PLUSW1,PORTD            ;move FSR1 offset by WREG
            BCF         PORTC,WE0,0             ;pulse the active-lo write
            BSF         PORTC,WE0,0             ; enable to capture data
            BSF         PORTC,CE0,0             ;set active-lo chip enable
            MOVLW       b'11111111'             ;PORT D I/O direct. template
            MOVWF       TRISD,0                 ;define PORT D I/O directions

            RETURN
;_____________________________________________________________________________
;Name:      write_mem_1
;Purpose:   To write the single byte of data that is passed to this routine
;           to the addressed memory location in chip 1.
;Passed:    A 1-byte value is passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Port D is configured as an output port, memory chip 1 is
;           configured for input and activated, the byte value to be written
;           is recovered from the stack and output onto the data bus through
;           port D, the data value is captured into memory chip 1 by pulsing
;           the active-low write enable line low, the memory chip is placed
;           back into power-down mode, and port D is returned to input port
;           status.
;Notes:     When this assembly routine is called by the C program the single
;           argument is pushed onto the top of the software stack and the
;           stack pointer is incremented to the next available empty stack
;           location. The argument value is recovered by accessing the memory
;           location of the current stack pointer decremented by one. This
;           is accomplished by first setting the working register WREG to a
;           value of minus one (0xFF) and using the indirect addressing
;           operation PLUSW1 that uses WREG as an offset to the current stack
;           pointer. The write enable line of the memory chip 1 is controlled
;           by bit 1 of port E which has been equated in the included file
;           TK_PIC18F458_DL.inc to "WE1". Similarily, the chip enable line is
;           controlled by bit 0 which has been equated to "CE1" and the
;           output enable line is controlled by bit 2 which has been equated
;           to "OE1". Registers TRISD, PORTE, and PORTD are special function
;           registers that are accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 102, 104, and 47. MPLAB C18 C Compiler
;           User's Guide pages 38 - 45.
```

```
write_mem_1
            MOVLW       b'00000000'             ;configure port D for output
            MOVWF       TRISD,0                 ; by modifying direc template
            BSF         PORTE,OE1,0             ;set active-lo output enable
            BCF         PORTE,CE1,0             ;clear active-lo chip enable
            MOVLW       h'FF'                   ;load -1 into WREG
            MOVFF       PLUSW1,PORTD            ;move FSR1 offset by WREG
            BCF         PORTE,WE1,0             ;pulse the active-lo write
            BSF         PORTE,WE1,0             ; enable to capture data
            BSF         PORTE,CE1,0             ;set active-lo chip enable
            MOVLW       b'11111111'             ;PORT D I/O direct. template
            MOVWF       TRISD,0                 ;define PORT D I/O directions

            RETURN
;_____________________________________________________________________________
;Name:      test_mem_clear
;Purpose:   To return a non-zero value if the memory is clear and preset.
;Passed:    No variables are passed to this routine.
;Returned:  One 8-bit value is returned from this routine.
;Actions:   Port D of the PIC18F458 microcontroller is configured for data
;           input. The memory address is set to 2^16 and the counting index
;           i, which has
;           been reserved as a 1-byte variable, is initialized to zero.
;           Next, the address 2^16 registers of memory chips 0 and 1 are read
;           and compared to the value of the counting index i. If the values
;           of the memory registers matches the index value, then the index
;           value and the address value are incremented and the process is
;           repeated for values of i up to and including 255. If all these
;           memory register values match the index values, then the memory is
;           assumed to be clear and preset and a nonzero value is returned.
;           If any memory register value does not match the associated index
;           value, then the memory is not clear and a zero value is returned.
;Notes:     8-bit arguments are transferred from assembly functions to C
;           programs in the WREG register. Registers LATD, CMCON, TRISD,
;           LATA, PORTC, PORTD, PORTE, PORTA, and WREG are special function
;           registers that are accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 102, 249, 93, 100, 104, 93, and 47.
;           MPLAB C18 C Compiler User's Guide pages 37.
test_mem_clear
;configuring port D for data input ------------------------------------
            CLRF        LATD,0                  ;clear output data latch
            MOVLW       b'00000111'             ;disable all comparator
            MOVWF       CMCON,0                 ; functions
            MOVLW       b'11111111'             ;PORT D I/O direct. template
            MOVWF       TRISD,0                 ;define PORT D I/O directions
;reset memory address to zero -----------------------------------------
            CLRF        LATA,0                  ;clear port A output (LSBs)
            BSF         PORTC,MR,0              ;generate a positive reset
            NOP                                 ;pulse on 74HC4040 master
            BCF         PORTC,MR,0              ;reset line (MSBs)
;increment memory address up to a value of 2^16 -----------------------
            CLRF        count_1,0               ;clear count_1 counter value
            CLRF        count_2,0               ;clear count_2 counter value
count_loop  INCF        PORTA,1,0               ;increment address value
            DECFSZ      count_1,1,0             ;decr count_1, count_1 = 0 ?
            GOTO        count_loop              ;(NO) go to count_loop
            DECFSZ      count_2,1,0             ;decr count_2, count_2 = 0 ?
```

```
            GOTO        count_loop              ;(NO) go to count_loop
;initializing counter index to zero ----------------------------------
            CLRF        i,0                     ;clear counting index regist.
;read memory 0 location ----------------------------------------------
cycle       BSF         PORTC,WE0,0             ;set active-lo write enable
            BCF         PORTC,CE0,0             ;clear active-lo chip enable
            BCF         PORTC,OE0,0             ;clear active-lo output enab
            MOVF        PORTD,0,0               ;move data at port D to WREG
            BSF         PORTC,OE0,0             ;set active_lo output enable
            BSF         PORTC,CE0,0             ;set active_lo chip enable
;compare memory 0 location to counting index --------------------------

            CPFSEQ      i,0                     ;is mem 0 value = i ?
            GOTO        notclear                ;(NO) go to notclear
;read memory 1 location ----------------------------------------------
            BSF         PORTE,WE1,0             ;set active-lo write enable
            BCF         PORTE,CE1,0             ;clear active-lo chip enable
            BCF         PORTE,OE1,0             ;clear active-lo output enab
            MOVF        PORTD,0,0               ;move data at port D to WREG
            BSF         PORTE,OE1,0             ;set active_lo output enable
            BSF         PORTE,CE1,0             ;set active_lo chip enable
;compare memory 1 location to counting index --------------------------
            CPFSEQ      i,0                     ;is mem 1 value = i ?
            GOTO        notclear                ;(NO) go to notclear
;increment memory address value ---------------------------------------
            INCF        PORTA,1,0               ;increment port A (LSBs)
;increment counting index and check for zero value --------------------
            INCFSZ      i,1,0                   ;increment i, is i=0 ?
            GOTO        cycle                   ;(NO) go to cycle
;memories are clear and preset, return nonzero value ------------------
            SETF        WREG,0                  ;set WREG to nonzero value
            RETURN
;memories are not clear, return zero value ----------------------------
notclear    CLRF        WREG,0                  ;NO, clear WREG

            RETURN
;_____________________________________________________________________________
;Name:      pre_conf_acq
;Purpose:   To preconfigure the data logger for data acquisition.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Additional INT0 interrupts are disabled by clearing INT0IE,
;           port D of the microcontroller is placed in a high-impedence mode
;           to avoid data bus collisions, both memory chips are enabled and
;           configured for data writing, and the ADC08062 analog to digital
;           converter is enabled.
;Notes:     Register LATD, CMCON, TRISD, PORTC, PORTE, and PORTB are special
;           function registers that is accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 102, 249, 100, 104, 96, and 47.

pre_conf_acq
;disable addition INT0 interrupts by clearing INT0IE in INTCON ---------
            BCF         INTCON,INT0IE,0         ;clear int enable flag
;ensure that microcontroller port D is in high-impedence input mode ----
            CLRF        LATD,0                  ;clear output data latch
            MOVLW       b'00000111'             ;disable all comparator
            MOVWF       CMCON,0                 ; functions
```

```
            MOVLW       b'11111111'             ;PORT D I/O direct. template
            MOVWF       TRISD,0                 ;define PORT D I/O directions
;configure memory chip 0 for data write -------------------------------
            BSF         PORTC,CE0,0             ;memory chip 0 is disabled
            BSF         PORTC,OE0,0             ;mem data lines set for input
            BCF         PORTC,WE0,0             ;writing to mem 0 is enabled
;configure memory chip 1 for data write -------------------------------
            BSF         PORTE,CE1,0             ;memory chip 1 is disabled
            BSF         PORTE,OE1,0             ;mem data lines set for input
            BCF         PORTE,WE1,0             ;writing to mem 1 is enabled
;configure ADC08062 analog to digital converter ------------------------
            BCF         PORTB,CS,0              ;enable anal to dig converter

            RETURN
;_____________________________________________________________________________
;Name:      get_data_0
;Purpose:   To trigger the acquisition of a single 8-bit data value from the
;           analog input line VIN1 of analog to digital converter ADC08062
;           and store the value to the current address on memory chip 0.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Analog input line VIN1 is selected, the conversion is initiated
;           and a time delay is generated to allow the conversion to proceed,
;           the converted value is output to the data bus, and this value is
;           captured in memory chip 0.
;Notes:     Registers PORTB and PORT C are special function registers that
;           are accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 96, 100, and 47.

get_data_0
            BCF         PORTB,A0,0              ;select VIN1 data for conver
            BCF         PORTB,WR,0              ;pulse write line low to
            BSF         PORTB,WR,0              ; initiate conversion
            NOP                                 ;time delay to allow the
            NOP                                 ; conversion to proceed
            NOP                                 ;
            NOP                                 ;
            BCF         PORTB,RD,0              ;output conv value on bus
            NOP                                 ;time delay for data output
            BCF         PORTC,CE0,0             ;capture conversion data in
            BSF         PORTC,CE0,0             ; memory chip 0
            BSF         PORTB,RD,0              ;remove conv data from bus


            RETURN
;_____________________________________________________________________________
;Name:      get_data_1
;Purpose:   To trigger the acquisition of a single 8-bit data value from the
;           analog input line VIN2 of analog to digital converter ADC08062
;           and store the value to the current address on memory chip 1.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Analog input line VIN2 is selected, the conversion is initiated
;           and a time delay is generated to allow the conversion to proceed,
;           the converted value is output to the data bus, and this value is
;           captured in memory chip 1.
;Notes:     Registers PORTB and PORT C are special function registers that
```

```
;           are accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 96, 100, and 47.

get_data_1
            BSF         PORTB,A0,0              ;select VIN2 data for conver
            BCF         PORTB,WR,0              ;pulse write line low to
            BSF         PORTB,WR,0              ; initiate conversion
            NOP                                 ;time delay to allow the
            NOP                                 ; conversion to proceed
            NOP                                 ;
            NOP                                 ;
            BCF         PORTB,RD,0              ;output conv value on bus
            NOP                                 ;time delay for data output
            BCF         PORTE,CE1,0             ;capture conversion data in
            BSF         PORTE,CE1,0             ; memory chip 1
            BSF         PORTB,RD,0              ;remove conv data from bus


            RETURN
;_____________________________________________________________________________
;Name:      post_conf_acq
;Purpose:   To postconfigure the data logger after the data acquisition.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Port D of the microcontroller is placed in a high-impedence mode
;           to avoid data bus collisions, both memory chips are disabled and
;           configured for data nonwriting, the ADC08062 analog to digital
;           converter is disabled, the interrupt flag is cleared, and
;           additional interrupts are enabled.
;Notes:     Register LATD, CMCON, TRISD, PORTC, PORTE, and PORTB are special
;           function registers that is accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 102, 249, 100, 104, 96, 79, and 47.

post_conf_acq
;ensure that microcontroller port D is in high-impedence input mode ----
            CLRF        LATD,0                  ;clear output data latch
            MOVLW       b'00000111'             ;disable all comparator
            MOVWF       CMCON,0                 ; functions
            MOVLW       b'11111111'             ;PORT D I/O direct. template
            MOVWF       TRISD,0                 ;define PORT D I/O directions
;configure memory chip 0 for data nonwrite ----------------------------
            BSF         PORTC,CE0,0             ;memory chip 0 is disabled
            BSF         PORTC,OE0,0             ;mem data lines set for input
            BSF         PORTC,WE0,0             ;writing to mem 0 is disabled
;configure memory chip 1 for data nonwrite ----------------------------
            BSF         PORTE,CE1,0             ;memory chip 1 is disabled
            BSF         PORTE,OE1,0             ;mem data lines set for input
            BSF         PORTE,WE1,0             ;writing to mem 1 is disabled
;configure ADC08062 analog to digital converter ------------------------
            BSF         PORTB,CS,0              ;disable anal to dig convert
;clear INT0 interrupt flag and enable additional interrupts ------------
            BCF         INTCON,INT0IF,0         ;clear interrupt flag
            BSF         INTCON,INT0IE,0         ;set int enable flag

            RETURN
;_____________________________________________________________________________
;Name:      time_delay
```

```
;Purpose:   To generate a time delay with a duration that is determined by
;           the values of the three passed arguments.
;Passed:    Three 1-byte values are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   The passed arguments are recovered from the software stack and
;           incremented by one. These three arguments are then used as the
;           counters for three nested loops with the inner loop counter
;           repeatedly being decremented to zero before the middle loop
;           counter is decremented to zero before the outer loop is
;           decremented. The time delay is terminated when all three counters
;           reach a value of zero.
;Notes:     When this assembly routine is called by the C program the three
;           arguments in the argument list are pushed onto the top of the
;           software stack in right-to-left order and the stack pointer is
;           incremented to the next available empty stack location. Thus,
;           after the function call the left-most argument in the argument
;           list is on the top of the software stack and the stack pointer
;           is pointing to the adjacent empty stack position. The left-most
;           arguement value is recovered by accessing the memory location of
;           the current stack pointer decremented by one. This is
;           accomplished by first setting the working register WREG to a
;           value of minus one (0xFF) and using the indirect addressing
;           operation PLUSW1 that uses WREG as an offset to the current stack
;           pointer. The second argument is recovered by setting WREG to
;           minus two (0xFE) and again using the indirect addressing
;           operation. The third argument is recovered by setting WREG to
;           minus three (0xFD) and again using the indirect addressing
;           operation. Each counter is decremented before it is tested for a
;           zero value. Therefore, each counter is incremented by one to
;           yield minimum delay intervals for counter values of 0 and maximum
;           delay intervals for counter values of 255. The duration of the
;           generated time delay is:
;
;             delay = (19 + 4*D1 + 1027*D2 + 262915*D3) machine cycles
;
;           where D1, D2, and D3 are the argument values as passed to the
;           function before being incremented and the period of a machine
;           cycle is four times the period of the microcontroller oscillator.
;           "delay_cnt_1", "delay_cnt_2", and "delay_cnt-3" are 1-byte
;           variable locations that have been reserved in access bank RAM at
;           the beginning of this assembly code file.
;References:PIC18FXX8 Data Sheet page 55. MPLAB C18 C Compiler User's Guide
;           pages 38 - 45.
time_delay
            MOVLW       h'FF'                   ;load -1 into WREG
            MOVFF       PLUSW1,delay_cnt_3      ;move FSR1 offset by WREG
            INCF        delay_cnt_1,1,0         ;incr value of delay_cnt_3
            MOVLW       h'FE'                   ;load -2 into WREG
            MOVFF       PLUSW1,delay_cnt_2      ;move FSR1 offset by WREG
            INCF        delay_cnt_2,1,0         ;incr value of delay_cnt_2
            MOVLW       h'FD'                   ;load -3 into WREG
            MOVFF       PLUSW1,delay_cnt_1      ;move FSR1 offset by WREG
            INCF        delay_cnt_3,1,0         ;incr value of delay_cnt_1
delay_loop  NOP
            DECFSZ      delay_cnt_1,1,0         ;decrement delay_cnt_1
            GOTO        delay_loop              ; until zero
            DECFSZ      delay_cnt_2,1,0         ;decrement delay_cnt_2
```

```
            GOTO        delay_loop              ; until zero
            DECFSZ      delay_cnt_3,1,0         ;decrement delay_cnt_3
            GOTO        delay_loop              ; until zero

            RETURN
;_____________________________________________________________________________
;Name:      turn_led_on
;Purpose:   To turn on the annuciator light emitting diode.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   The bit that powers the LED annunciator is set high.
;Notes:     It is assumed that the anode of the LED is connected to output
;           bit 0 of port C and the cathode of the LED is connected to
;           ground. This bit has been equated in the included file
;           TK_PIC18F458_DL.inc to "LED". Register PORTC is a special
;           function registers that is accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 100 and 47.

turn_led_on
            BSF         PORTC,LED,0             ;turn on led annunciator

            RETURN
;_____________________________________________________________________________
;Name:      turn_led_off
;Purpose:   To turn off the annuciator light emitting diode.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   The bit that powers the LED annunciator is cleared low.
;Notes:     It is assumed that the anode of the LED is connected to output
;           bit 0 of port C and the cathode of the LED is connected to
;           ground. This bit has been equated in the included file
;           TK_PIC18F458_DL.inc to "LED". Register PORTC is a special
;           function register that is accessed through the access bank.
;References:PIC18FXX8 Data Sheet pages 100 and 47.

turn_led_off
            BCF         PORTC,LED,0             ;turn on led annunciator

            RETURN
;_____________________________________________________________________________
;Name:      clear_int_flag
;Purpose:   To clear the INT0 interrupt flag.
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:   Bit INT0IF in register INTCON is cleared.
;Notes:     Register INTCON is a special function register that is accessed
;           through the access bank.
;References:PIC18FXX8 Data Sheet pages 92 and 47.

clear_int_flag
            BCF         INTCON,INT0IF,0         ;clear interrupt flag
            BSF         INTCON,INT0IE,0         ;set int enable flag

            RETURN
;_____________________________________________________________________________
;Name:      acquire_data
;Purpose:   To acquire all data.
```

```
;Passed:    No variables are passed to this routine.
;Returned:  No variables are returned from this routine.
;Actions:
;Notes:     Register INTCON is a special function register that is accessed
;           through the access bank.
;References:PIC18FXX8 Data Sheet pages 92 and 47.

acquire_data
;FILLING THE FIRST HALF OF MEMORY WITH DATA ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;the following code segment acquires one data value from channel 0 ----------
data_loop_1 BCF         PORTB,A0,0              ;select VIN1 data for conver
            BCF         PORTB,WR,0              ;pulse write line low to
            BSF         PORTB,WR,0              ; initiate conversion
            INCF        PORTA,1,0               ;increment port A (LSBs)
            NOP                                 ;time delay to allow the
            NOP                                 ; conversion to proceed
            ;
            BCF         PORTB,RD,0              ;output conv value on bus
            BCF         PORTC,CE0,0             ;capture conversion data in
            BSF         PORTC,CE0,0             ; memory chip 0
            BSF         PORTB,RD,0              ;remove conv data from bus
;the following code segment acquires one data value from channel 1 ----------
            BSF         PORTB,A0,0              ;select VIN2 data for conver
            BCF         PORTB,WR,0              ;pulse write line low to
            BSF         PORTB,WR,0              ; initiate conversion
            NOP                                 ;time delay to allow the
            NOP                                 ; conversion to proceed
            NOP                                 ;
            BCF         PORTB,RD,0              ;output conv value on bus
            BCF         PORTE,CE1,0             ;capture conversion data in
            BSF         PORTE,CE1,0             ; memory chip 1
            BSF         PORTB,RD,0              ;remove conv data from bus


            BTFSS       PORTC,AMSB,0            ;is first half of mem. full?
            GOTO        data_loop_1             ;(NO) go to data_loop_1

;FILLING THE SECOND HALF OF MEMORY WITH DATA
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;the following code segment acquires one data value from channel 0 ----------
data_loop_2 BCF         PORTB,A0,0              ;select VIN2 data for conver
            BCF         PORTB,WR,0              ;pulse write line low to
            BSF         PORTB,WR,0              ; initiate conversion
            INCF        PORTA,1,0               ;increment port A (LSBs)
            NOP                                 ;time delay to allow the
            NOP                                 ; conversion to proceed
            ;
            BCF         PORTB,RD,0              ;output conv value on bus
            BCF         PORTC,CE0,0             ;capture conversion data in
            BSF         PORTC,CE0,0             ; memory chip 0
            BSF         PORTB,RD,0              ;remove conv data from bus
;the following code segment acquires one data value from channel 1 ----------
            BSF         PORTB,A0,0              ;select VIN2 data for conver
            BCF         PORTB,WR,0              ;pulse write line low to
            BSF         PORTB,WR,0              ; initiate conversion
            NOP                                 ;time delay to allow the
            NOP                                 ; conversion to proceed
```

```
            NOP                                 ;
            BCF         PORTB,RD,0              ;output conv value on bus
            BCF         PORTE,CE1,0             ;capture conversion data in
            BSF         PORTE,CE1,0             ; memory chip 1
            BSF         PORTB,RD,0              ;remove conv data from bus


            BTFSC       PORTC,AMSB,0            ;is first half of mem. full?
            GOTO        data_loop_2             ;(NO) go to data_loop_1

            RETURN
;_____________________________________________________________________________
;Name:      test_interrupt
;Purpose:   To return a return a non-zero value if the RB0/INT0 line is
;           pulled high.
;Passed:    No variables are passed to this routine.
;Returned:  One 8-bit value is returned from this routine.
;Actions:   The contents of the serial port receive register, RCREC, are
;           moved to the working register, WREG, and the serial port receive
;           interrupt flag bit, RCIF, of the peripheral interrupt request
;           register number one, PIR1, is cleared.
;Notes:     8-bit arguments are transferred from assembly functions to C
;           programs in the WREG register. Registers PIR1 and RCREG are
;           special function registers that are accessed through the access
;           bank.
;References:PIC18FXX8 Data Sheet pages 191, 82 and 47. MPLAB C18 C Compiler
;           User's Guide pages 37.
test_interrupt
            BTFSS       PORTB,0,0               ;is RB0/INT0 set?
            GOTO        rb0_not_set             ;(NO) go to rb0_not_set
            SETF        WREG,0                  ;set W
            RETURN                              ;return a non-zero value
rb0_not_set CLRF        WREG,0                  ;clear W
            RETURN                              ;return a zero value

            end                                 ;end of default code segment
```

INTENTIONALLY LEFT BLANK.

Appendix C. Listing of Ballistic Data Logger Microcontroller Assembler Header File

This appendix appears in its original form, without editorial change.

```
;Template file for Microchip PIC18F458 microcontroller used in IR Data Logger
;Version 1.0
;April 2006

;Define Processor       ***************************************************

            processor   18F458

;Define Special Function Register locations     ****************************

TOSU        equ         h'FFF'                  ;Top Of Stack Upper     pg 38
TOSH        equ         h'FFE'                  ;Top Of Stack High      pg 38
TOSL        equ         h'FFD'                  ;Top Of Stack Low pg 38

STKPTR      equ         h'FFC'                  ;STacK PoinTeR    pg 38
STKFUL      equ         h'7'                    ;stack full flag, 1=full
STKUNF      equ         h'6'                    ;stack underflow, 1=underflow

PCLATU      equ         h'FFB'                  ;Prog Count Latch Upper pg 40
PCLATH      equ         h'FFA'                  ;Prog Count Latch High  pg 40
PCL         equ         h'FF9'                  ;Program Counter Low    pg 40
TBLPTRU     equ         h'FF8'                  ;TaBLe PoinTeR Upper    pg 65
TBLPTRH     equ         h'FF7'                  ;TaBLe PoinTeR High     pg 65
TBLPTRL     equ         h'FF6'                  ;TaBLe PoinTeR Low      pg 65
TABLAT      equ         h'FF5'                  ;TABle LATch      pg 65
PRODH       equ         h'FF4'                  ;PRODuct High     pg 75
PRODL       equ         h'FF3'                  ;PRODuct Low      pg 75

INTCON      equ         h'FF2'                  ;Interrupt Config pg 79
GIE_GIEH    equ         h'7'                    ;global int enable/high enab
PEIE_GIEL   equ         h'6'                    ;peri int en/glob in enab low
TMR0IE      equ         h'5'                    ;timer 0 overflow inter enab
INT0IE      equ         h'4'                    ;INT0 external interrupt enab
RBIE        equ         h'3'                    ;port B change interrupt enab
TMR0IF      equ         h'2'                    ;timer 0 overflow inter flag
INT0IF      equ         h'1'                    ;INT0 external interrupt flag
RBIF        equ         h'0'                    ;port B change interrupt flag

INTCON2     equ         h'FF1'                  ;Interrupt Config 2     pg 80
RBPU_       equ         h'7'                    ;port B pull-ups enable, 0=en
INTEDG0     equ         h'6'                    ;ext interrupt 0 edge select
INTEDG1     equ         h'5'                    ;ext enterrupt 1 edge select
TMR0IP      equ         h'2'                    ;timer 0 overflow int prior.
RBIP        equ         h'0'                    ;port B change inter priority

INTCON3     equ         h'FF0'                  ;Interrupt Config 3     pg 81
INT2IP      equ         h'7'                    ;INT2 external inter priority
INT1IP      equ         h'6'                    ;INT1 external inter priority
INT2IE      equ         h'4'                    ;INT2 external inter enable
INT1IE      equ         h'3'                    ;INT1 external inter enable
INT2IF      equ         h'1'                    ;INT2 external inter flag
INT1IF      equ         h'0'                    ;INT1 external inter flag
```

```
INDF0       equ         h'FEF'              ;INDirect File 0  pg 55
POSTINC0    equ         h'FEE'              ;POST INCrement 0 pg 55
POSTDEC0    equ         h'FED'              ;POST DECrement 0 pg 55
PREINC0     equ         h'FEC'              ;PRE INCrement 0  pg 55
PLUSW0      equ         h'FEB'              ;PLUS Wreg 0      pg 55
FSR0H       equ         h'FEA'              ;File Select Reg High 0 pg 55
FSR0L       equ         h'FE9'              ;File Select Reg Low 0  pg 55
WREG        equ         h'FE8'              ;Working REGister
INDF1       equ         h'FE7'              ;INDirect File 1  pg 55
POSTINC1    equ         h'FE6'              ;POST INCrement 1 pg 55
POSTDEC1    equ         h'FE5'              ;POST DECrement 1 pg 55
PREINC1     equ         h'FE4'              ;PRE INCrement 1  pg 55
PLUSW1      equ         h'FE3'              ;PLUS Wreg 1      pg 55
FSR1H       equ         h'FE2'              ;File Select Reg High 1 pg 55
FSR1L       equ         h'FE1'              ;File Select Reg Low 1  pg 55
BSR         equ         h'FE0'              ;Bank Select Register   pg 54
INDF2       equ         h'FDF'              ;INDirect File 2  pg 55
POSTINC2    equ         h'FDE'              ;POST INCrement 2 pg 55
POSTDEC2    equ         h'FDD'              ;POST DECrement 2 pg 55
PREINC2     equ         h'FDC'              ;PRE INCrement 2  pg 55
PLUSW2      equ         h'FDB'              ;PLUS Wreg 2      pg 55
FSR2H       equ         h'FDA'              ;File Select Reg High 2 pg 55
FSR2L       equ         h'FD9'              ;File Select Reg Low 2  pg 55

STATUS      equ         h'FD8'              ;arithmetic STATUS reg  pg 57
N           equ         h'4'                ;negative bit
OV          equ         h'3'                ;overflow bit
Z           equ         h'2'                ;zero bit
DC          equ         h'1'                ;digit carry/borrow bit
C           equ         h'0'                ;carry/borrow bit

TMR0H       equ         h'FD7'              ;TiMeR 0 High          pg 111
TMR0L       equ         h'FD6'              ;TiMeR 0 Low           pg 111

T0CON       equ         h'FD5'              ;Timer 0 CONfiguration pg 109
TMR0ON      equ         h'7'                ;timer 0 on/off control bit
T08BIT      equ         h'6'                ;timer 0 8/16 bit control bit
T0CS        equ         h'5'                ;timer 0 clock source bit
T0SE        equ         h'4'                ;timer 0 edge select bit
PSA         equ         h'3'                ;timer 0 prescalar desel bit
T0PS2       equ         h'2'                ;timer 0 prescal scale bit 2
T0PS1       equ         h'1'                ;timer 0 prescal scale bit 1
T0PS0       equ         h'0'                ;timer 0 prescal scale bit 0

OSCCON      equ         h'FD3'              ;OSCillator CONfig      pg 20
SCS         equ         h'0'                ;system clock switch bit

LVDCON      equ         h'FD2'              ;Low Volt Dett CONfig  pg 261
IRVST       equ         h'5'                ;low voltage interrupt enable
LVDEN       equ         h'4'                ;low voltage device enable
LVDL3       equ         h'3'                ;low volt detect level bit 3
LVDL2       equ         h'2'                ;low volt detect level bit 2
LVDL1       equ         h'1'                ;low volt detect level bit 1
LVDL0       equ         h'0'                ;low volt detect level bit 0

WDTCON      equ         h'FD1'              ;Watch Dog Time CONfig pg 272
```

```
SWDTEN      equ         h'0'                ;software controlled WDT enab

RCON        equ         h'FD0'              ;Reset CONfiguration    pg 58
IPEN        equ         h'7'                ;interrupt prior. enable bit
RI_         equ         h'4'                ;reset instruction flag
TO_         equ         h'3'                ;watch dog time out flag
PD_         equ         h'2'                ;power down detection flag
POR_        equ         h'1'                ;power on reset status bit
BOR_        equ         h'0'                ;brown out reset status bit

TMR1H       equ         h'FCF'              ;TiMeR 1 High          pg 113
TMR1L       equ         h'FCE'              ;TiMeR 1 Low           pg 113

T1CON       equ         h'FCD'              ;Timer 1 CONfig        pg 113
RD16        equ         h'7'                ;16 bit read/write mode enab
T1CKPS1     equ         h'5'                ;prescalar value bit 1
T1CKPS0     equ         h'4'                ;prescalar value
T1OSCEN     equ         h'3'                ;timer 1 oscillator enab bit
T1SYNC_     equ         h'2'                ;timer 1 ext clock synch bit
TMR1CS      equ         h'1'                ;timer 1 clock source select
TMR1ON      equ         h'0'                ;timer 1 on bit

TMR2        equ         h'FCC'              ;TiMeR 2               pg 117
PR2         equ         h'FCB'              ;Period Reg timer 2    pg 118

T2CON       equ         h'FCA'              ;Timer 2 CONfiguration pg 117
TOUTPS3     equ         h'6'                ;timer 2 post scal sel bit 3
TOUTPS2     equ         h'5'                ;timer 2 post scal sel bit 2
TOUTPS1     equ         h'4'                ;timer 2 post scal sel bit 1
TOUTPS0     equ         h'3'                ;timer 2 post scal sel bit 0
TMR2ON      equ         h'2'                ;timer 2 on bit
T2CKPS1     equ         h'1'                ;timer 2 clock prescale bit 1
T2CKPS0     equ         h'0'                ;timer 2 clock prescale bit 0

ADRESH      equ         h'FC4'              ;Anal/Dig Result High  pg 243
ADRESL      equ         h'FC3'              ;Anal/Dig Result Low   pg 243
ADCON0      equ         h'FC2'              ;Anal/Dig Config 0     pg 241
ADCON1      equ         h'FC1'              ;Anal/Dig Config 1     pg 242
CCPR1H      equ         h'FBF'              ;Capt/Comp/Pwm R 1 Hi  pg 127
CCPR1L      equ         h'FBE'              ;Capt/Comp/Pwm R 1 Low pg 127
CCP1CON     equ         h'FBD'              ;Capt/Comp/Pwm CONf 1  pg 123
ECCPR1H     equ         h'FBC'              ;Enhanced CCP Reg 1 Hi pg 133
ECCPR1L     equ         h'FBB'              ;Enhanced CCP Reg 1 Lo pg 133
ECCP1CON    equ         h'FBA'              ;Enhanced CCP 1 Config pg 131
ECCP1DEL    equ         h'FB7'              ;Enhanced CCP 1 Delay  pg 140
ECCPAS      equ         h'FB6'              ;Enh CCP Auto Shutdown pg 142
CVRCON      equ         h'FB5'              ;Comparator Volt Ref   pg 255
CMCON       equ         h'FB4'              ;Comparator Mod Config pg 249
TMR3H       equ         h'FB3'              ;TiMeR 3 High          pg 119
TMR3L       equ         h'FB2'              ;TiMeR 3 Low           pg 119
T3CON       equ         h'FB1'              ;Timer 3 CONfiguration pg 119
SPBRG       equ         h'FAF'              ;Ser Port Baud Rate Gn pg 185
RCREG       equ         h'FAE'              ;ReCeive Reg Ser Port  pg 191
TXREG       equ         h'FAD'              ;Transmit Reg Ser Port pg 189

TXSTA       equ         h'FAC'              ;Transmit STAtus reg   pg 183
TRMT        equ         h'1'                ;TSR status bit
```

```
RCSTA       equ         h'FAB'              ;ReCeive STAtus reg    pg 184
EEADR       equ         h'FA9'              ;Elec Erase Address     pg 59
EEDATA      equ         h'FA8'              ;Elec Erase Data  pg 59
EECON2      equ         h'FA7'              ;Elec Erase Config 2    pg 59
EECON1      equ         h'FA6'              ;Elec Erase Config 1    pg 60
IPR3        equ         h'FA5'              ;Interrupt Prior Reg 3  pg 90
PIR3        equ         h'FA4'              ;Peripheral Int Reg 3   pg 84
PIE3        equ         h'FA3'              ;Periph Int Enable 3    pg 87
IPR2        equ         h'FA2'              ;Interrupt Prior Reg 2  pg 89
PIR2        equ         h'FA1'              ;Peripheral Int Reg 2   pg 83
PIE2        equ         h'FA0'              ;Periph Int Enable 2    pg 86

IPR1        equ         h'F9F'              ;Interrupt Prior Reg 1  pg 88
PSPIP       equ         h'7'                ;Par slave port interr prior
ADIP        equ         h'6'                ;ADC interrupt priority
RCIP        equ         h'5'                ;USART rx interrupt priority
TXIP        equ         h'4'                ;USART tx interrupt priority
SSPIP       equ         h'3'                ;Master synch ser port prior
CCP1IP      equ         h'2'                ;CCP1 interrupt priority
TMR2IP      equ         h'1'                ;TMR2 to PR2 match priority
TMR1IP      equ         h'0'                ;TMR1 overflow inter prior

PIR1        equ         h'F9E'              ;Peripheral Int Reg 1   pg 82
RCIF        equ         h'5'                ;USART rec interr flag bit
TXIF        equ         h'4'                ;USART trans interr flag bit
TMR2IF      equ         h'1'                ;T2 to PR2 match int flag bit

PIE1        equ         h'F9D'              ;Periph Int Enable 1    pg 85
RCIE        equ         h'5'                ;USART rx interrupt enable
TMR2IE      equ         h'1'                ;TMR2 to PR2 match int enable

TRISE       equ         h'F96'              ;TRIState Port E       pg 104
TRISD       equ         h'F95'              ;TRIState Port D       pg 102
TRISC       equ         h'F94'              ;TRIState Port C       pg 100
TRISB       equ         h'F93'              ;TRIState Port B       pg 96
TRISA       equ         h'F92'              ;TRIState Port A       pg 93
LATE        equ         h'F8D'              ;LATch Port E          pg 104
LATD        equ         h'F8C'              ;LATch Port D          pg 102
LATC        equ         h'F8B'              ;LATch Port C          pg 100
LATB        equ         h'F8A'              ;LATch Port B          pg 96
LATA        equ         h'F89'              ;LATch Port A          pg 93

PORTE       equ         h'F84'              ;PORT E                pg 104
CE1         equ         h'0'                ;chip enab line for mem IC 1
WE1         equ         h'1'                ;write enab line for mem IC 1
OE1         equ         h'2'                ;output enab line mem IC 1


PORTD       equ         h'F83'              ;PORT D                pg 102

PORTC       equ         h'F82'              ;PORT C                pg 100
LED         equ         h'0'                ;light emitting diode power
AMSB        equ         h'1'                ;address most significant bit
CE0         equ         h'2'                ;chip enab line for mem IC 0
WE0         equ         h'3'                ;write enab line for mem IC 0
OE0         equ         h'4'                ;output enab line mem IC 0
```

```
MR          equ         h'5'                    ;74HC4040 master reset line
TX_BIT      equ         h'6'                    ;serial transmition bit
RX_BIT      equ         h'7'                    ;serial reception bit

PORTB       equ         h'F81'                  ;PORT B           pg 96
A0          equ         h'1'                    ;ADC08062 channel select bit
CS          equ         h'2'                    ;ADC08062 chip select
RD          equ         h'3'                    ;ADC08062 read line
WR          equ         h'4'                    ;ADC08062 write line

PORTA       equ         h'F80'                  ;PORT A           pg 93

;Set Processor Configuration Bits   ************************************

            __config    h'300001',b'00100101'   ;oscillator configuration
                                                ;pages 266, 19, 20
            __config    h'300002',b'0000100'    ;brown-out and power-up conf
                                                ;pages 266, 26
            __config    h'300003',b'00001110'   ;watch dog timer config
                                                ;pages 267, 272
            __config    h'300006',b'00000001'   ;debug,low-V ICSP,stack conf
                                                ;pages 267, 279
            __config    h'300008',b'00001111'   ;code protection config
                                                ;page 268
            __config    h'300009',b'11000000'   ;EEPROM, boot code prot conf
                                                ;page 268
            __config    h'30000A',b'00001111'   ;write protection config
                                                ;page 269
            __config    h'30000B',b'11100000'   ;EE, boot, con reg
                                                ;page 269
            __config    h'30000C',b'00001111'   ;table read protection
                                                ;page 270
            __config    h'30000D',b'01000000'   ;boot block table read prot
                                                ;page 270
```

Appendix D. Ballistic Data Logger Microcontroller Linker Script File

This appendix appears in its original form, without editorial change.

```
// $Id: 18f458i.lkr,v 1.3 2003/03/13 05:02:23 sealep Exp $
// File: 18f458i.lkr
// Sample linker script for the PIC18F458 processor

// Modified for Ballistic Data Logger

// File:                TK_18F458_DL.lkr
// Author:              Tom Kottke
// Date:                01 April 2006

// Library and object files which do not have a path are searched using the
// current directory
LIBPATH .

// MPLAB C18 start-up file c018i.o is specified for linking
FILES c018i.o
// MPLAB C18 standard library file is specified for linking
FILES clib.lib
// MPLAB C18 processor-specific library file is specified for linking
FILES p18f458.lib

// Definition of reset vector ROM memory region which is protected
CODEPAGE   NAME=reset_vector        START=0x0000      END=0x0007  PROTECTED
// Definition of high priority interrupt vector ROM memory region, protected
CODEPAGE   NAME=hi_int_vector       START=0x0008      END=0x0017  PROTECTED
// Definition of low priority interrupt vector ROM memory region, protected
CODEPAGE   NAME=lo_int_vector       START=0x0018      END=0x00FF  PROTECTED
// Definition of main ROM memory region where compiled code can be located
CODEPAGE   NAME=main_vector         START=0x0100      END=0x7DBF
// Definition of resource memory region used by ICD2 in debug mode, protected
CODEPAGE   NAME=debug      START=0x7DC0         END=0x7FFF         PROTECTED
// Definition of microcontroller ID location ROM memory region, protected
CODEPAGE   NAME=idlocs     START=0x200000       END=0x200007       PROTECTED
// Definition of configuration register ROM memory region which is protected
CODEPAGE   NAME=config     START=0x300000       END=0x30000D       PROTECTED
//Definition of device ID location ROM memory region which is protected
CODEPAGE   NAME=devid      START=0x3FFFFE       END=0x3FFFFF       PROTECTED

//Definition of RAM memory regions
ACCESSBANK NAME=accessram  START=0x0            END=0x5F
DATABANK   NAME=gpr0       START=0x60           END=0xFF
DATABANK   NAME=gpr1       START=0x100          END=0x1FF
DATABANK   NAME=gpr2       START=0x200          END=0x2FF
DATABANK   NAME=gpr3       START=0x300          END=0x3FF
DATABANK   NAME=gpr4       START=0x400          END=0x4FF
DATABANK   NAME=gpr5       START=0x500          END=0x5F3
DATABANK   NAME=dbgspr     START=0x5F4          END=0x5FF          PROTECTED
DATABANK   NAME=bankedsfr  START=0xF00          END=0xF5F          PROTECTED
ACCESSBANK NAME=accesssfr  START=0xF60          END=0xFFF          PROTECTED

//Definition of logical sections
SECTION    NAME=ACCESSRAM       RAM=accessram
```

```
SECTION    NAME=hi_int_vector   ROM=hi_int_vector
SECTION    NAME=main_vector     ROM=main_vector
SECTION    NAME=CONFIG          ROM=config

//Definition of software stack memory region
STACK SIZE=0xF3 RAM=gpr5
```

NO. OF
COPIES ORGANIZATION

1 (PDF ONLY) DEFENSE TECHNICAL INFORMATION CTR
DTIC OCA
8725 JOHN J KINGMAN RD
STE 0944
FORT BELVOIR VA 22060-6218

1 US ARMY RSRCH DEV & ENGRG CMD
SYSTEMS OF SYSTEMS INTEGRATION
AMSRD SS T
6000 6TH ST STE 100
FORT BELVOIR VA 22060-5608

1 INST FOR ADVNCD TCHNLGY
THE UNIV OF TEXAS
AT AUSTIN
3925 W BRAKER LN
AUSTIN TX 78759-5316

1 DIRECTOR
US ARMY RESEARCH LAB
IMNE ALC IMS
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CI OK TL
2800 POWDER MILL RD
ADELPHI MD 20783-1197

ABERDEEN PROVING GROUND

1 DIR USARL
AMSRD ARL CI OK TP (BLDG 4600)

NO. OF COPIES	ORGANIZATION
42	DIR USARL
	AMSRD ARL WM BA
	D LYON
	AMSRD ARL WM BC
	I CELMINS
	P PLOSTINS
	T PUCKETT
	AMSRD ARL WM BD
	B FORCH
	AMSRD ARL WM BF
	D WILKERSON
	AMSRD ARL WM MA
	M VANLANDINGHAM
	AMSRD ARL WM MB
	L BURTON
	AMSRD ARL WM MC
	M MAHER
	AMSRD ARL WM MD
	E CHIN
	AMSRD ARL WM TA
	J BALL
	T BARNHILL
	M BURKINS
	J RAYMOND
	D SCHALL
	S SCHOENFELD
	R STRICKLAND
	AMSRD ARL WM TB
	T ADKINS
	D PILARSKI
	R SKAGGS
	AMSRD ARL WM TC
	R COATES
	E DEAL
	J KOONTZ
	D WEEKS
	AMSRD ARL WM TD
	T BJERKE
	AMSRD ARL WM TE
	P BERNING
	J CAMERON
	C HUMMER
	S KENNEDY
	T KOTTKE
	M MCNEIR
	K MAHAN
	A NIILER
	J POWELL
	B RINGERS
	G THOMSON

NO. OF COPIES	ORGANIZATION
	CSTE DTC AT TT I
	G BARTLETT
	K DICK
	G HETTCHEN
	J PHILISTINE
	CSTE DTC AT SL
	D VALZ
	CSTE DTC AT SL B
	M CLARK

INTENTIONALLY LEFT BLANK.